Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Уральский государственный педагогический университет»
Институт физики, технологии и экономики
Кафедра физики и математического моделирования

Проектирование и разработка корпоративной информационной системы для электронных услуг потребителей

Выпускная квалификационная работа

Квалификационная работа
допущена к защите
Зав. кафедрой
д.ф-м.н., проф. Сидоров В.Е.

_____ _____
дата подпись

Исполнитель:
Лукьяненко Валерий Витальевич,
студент IV курса,
группы БЭ-41z

_____
подпись

Научный руководитель:
Стихина Наталья Владимировна,
к.п.н., доцент

_____
подпись

Екатеринбург – 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Роль интернета в жизни современного человека сложно переоценить. По данным международного агентства «We are social» 42% населения земли регулярно используют интернет [1].

За внешней частью глобальной сети, за ее удобством, свободой и доступностью скрывается работа многих тысяч людей. Развитие информационных технологий, концепций создания программного обеспечения и постоянная эволюция элементной базы позволяют ускорить разработку веб-решений, повышая их качество, удобство и отказоустойчивость.

В настоящее время веб-разработка - это одна из самых динамично развивающихся и меняющихся областей IT. Количество всевозможных стандартов, языков, библиотек и актуальных навыков в ней растет огромными темпами.

У современных разработчиков появилась возможность создавать веб-приложения нового поколения. Сегодняшний Интернет является результатом непрерывных усилий открытого веб-сообщества, которое помогает разрабатывать такие технологии, как HTML 5, CSS3 и WebGL, и добивается их поддержки всеми браузерами.

Цель данной работы – проектирование и разработка корпоративной информационной системы для записи детей в детские сады.

Данная проблема является актуальной, так как изменение в демографической ситуации привело к потребности увеличить число детских садов. Повышение рождаемости привело к большим очередям и нехватке мест в муниципальных детских садах. Альтернативой муниципальным учреждениям послужили частные детские сады. Но возникла проблема поиска подходящего дошкольного учреждения. В данный момент отсутствуют современные сервисы, предоставляющие полный набор услуг, связанных с поиском учреждений дошкольного образования по различным критериям и возможностью за-

писи в них.

Данный проект является социально значимым, предоставляя актуальную информацию и современные услуги, сервис способствует развитию отрасли частных дошкольных учреждений.

Для достижения поставленной цели сформулируем следующие задачи:

1) проанализировать предметную область;

2) спроектировать веб-приложение;

3) создать дизайн-концепцию и осуществить верстку макетов страниц;

4) разработать функциональные инструменты и интегрировать в систему управления содержимым;

5) разместить материалы на сайте и произвести пробную эксплуатацию.

1. ТЕХНОЛОГИИ РАЗРАБОТКИ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1. Структура веб-технологий

Благодаря Интернету пользователи получили возможность свободного обмена информацией. Ранее Интернет позволял обмениваться только файлами и обычным текстом, при том без форматирования. Лишь в конце 80-х гг. появилась возможность обмена информацией любого типа. Язык разметки HTML (Hyper Text Markup Language), универсальный идентификатор ресурса URL (Uniform Resource Locator) и протокол обмена гипертекстовой информацией HTTP (HyperText Transfer Protocol) стали основными компонентами Всемирной паутины [2].

Сеть World Wide Web представляет собой глобальное информационное пространство, основанное на физической инфраструктуре Интернета и протоколе передачи данных HTTP. Зачастую, говоря об Интернете, подразумевают именно сеть Веб.

Объяснением популярности сети Веб служит тот факт, что даже новые пользователи с легкостью используют ее благодаря применению графических интерфейсов. Всемирная паутина обладает огромным количеством информации, а ее использование способно дать ответ на практически любой вопрос.

Всемирная паутина берет свое начало в 1989 году в Европейском центре ядерных исследований CERN (Conseil Europeen pour la Recherche Nucleaire) в Швейцарии. Основой для создания служила потребность обмена информацией между учеными, которые жили в разных странах. Им необходимо было обмениваться рабочими документами, в том числе фотографиями и чертежами. Физик CERN Тим Бернерс-Ли предложил создать сеть из связанных документов. Публичная демонстрация этой идеи была проведена на

конференции Hyertext’91 в Сан-Антонио в 1991 г. В этом же году Марк Андрессен из университета Иллинойса начал разработку первого браузера Mosaic, обладавшего графическим интерфейсом [3].

На протяжении 1990-х и 2000-х годов Всемирная сеть росла огромными темпами, постоянно увеличивалось количество веб-сайтов, пока их значение не перевалило за миллионы. Некоторые сайты приобрели огромную популярность. Сегодня компании, стоящие за этими сайтами и сервисами в большей мере определяют веб в том виде, в котором видим его сейчас.

В качестве примеров можно привести: магазин Amazon, основанный в 1994 году, рынок eBay (1995 г.), поисковик Google (1998 г.) и социальную сеть Facebook (2004 г.) [4].

В 1994 году Европейский центр ядерных исследований CERN совместно с Массачусетским технологическим институтом основали WWW-консорциум (World Wide Web Consortium, сокращенно W3C), целью которой является развитие Всемирной паутины, принятие стандартов, протоколов и развитие взаимодействия между сайтами.

Пользователи воспринимают Всемирную паутину как совокупность контента, распределенного по веб-страницам. Страница может ссылаться (указывать) на другие страницы, расположенные на веб-серверах в любой точке мира. Переходя по ссылке, страница загружается и отображается пользователю в браузере. При помощи ссылок пользователь может путешествовать по страницам веб-сайтов бесконечно.

Для просмотра страниц пользователь использует специальную программу, называемую браузер. Самыми популярными браузерами являются программы Firefox, Internet Exlorer и Chrome. Браузер предоставляет страницу, запрошенную пользователем, интерпретирует ее контент, форматирует и выводит на экран пользователя. Контентом является совокупность текста, изображений и видео, может выглядеть как простой документ или как программа с графическим интерфейсом.

Страницы могут быть связаны с другими страницами сайта при помощи гиперссылок. Гиперссылка – это ссылка на другую станицу, представляет собой строку текста, изображение, кнопку, значок и т.д. Переход по ссылке сообщает браузеру о том, что необходимо загрузить другую страницу и указывает адрес на нее.

Основной принцип, стоящий за отображением страниц.

Браузер формирует веб-страницу на машине пользователя. Для этого отсылается запрос на один или более веб-серверов, которые отвечая передают контент для формирования страницы. Протоколом запрос-ответа для отображения страницы служит простой текстовый протокол, работающий через TCP (transmission control protocol – протокол управления передачей). Этот протокол называется HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) [5]. Контентом для веб-страницы может быть документ, результат работы программы или запрос, отосланный в базу данных.

Если страница содержит статический документ, то она называется статичной. Если страница формируется по требованию программы или содержит программу, то она называется динамической.

Отображение страниц на клиентской машине

Браузер — это программа, отображающая веб-страницы, обладает графическим интерфейсом и способна распознавать щелчки мыши на активных элементах страницы. При выборе активного элемента браузер следует по адресу в гиперссылке, запрашивает у сервера страницу и получает ответ от сервера в виде запрашиваемой страницы.

Каждая страница содержит URL (Uniform Resource Locator — унифицированный указатель информационного ресурса), служащий именем страницы во Всемирной паутине. URL состоит из трех частей: протокол (который также называют схемой), DNS-имя машины, на которой расположена страница, и путь, уникально определяющий отдельную страницу (файл для чтения или программу, предназначенную для запуска на машине). В общем слу-

чае у пути есть иерархическое имя, которое моделирует структуру каталогов файлов. Однако интерпретация пути — это работа сервера. Действительная структура каталогов может и не отображаться.

Когда пользователь щелкает мышью на гиперссылке, браузером выполняется ряд действий, приводящих к загрузке страницы, на которую указывает ссылка. Опишем каждое действие, происходящее после выбора этой ссылки [6].

1) браузер определяет URL (по выбранному элементу страницы);

2) браузер запрашивает у службы DNS IP-адрес сервера;

3) DNS дает ответ, например: 128.208.3.88;

4) браузер устанавливает TCP-соединение с 80-м портом (общеизвестным портом для HTTP-протокола) машины 128.208.3.88;

5) браузер отправляет HTTP-запрос на получение файла */index.html*;

6) сервер высылает страницу, как HTTP-ответ, например, отправляя файл */index.html*;

7) если страница содержит URL, которые необходимы для отображения, браузер получает другие URL, используя тот же процесс. В этом случае URL включают множество размещенных изображений, также полученных с сервера, размещенное видео и скрипты;

8) браузер отображает страницу */index.html*;

9) если в течение некоторого времени на те же серверы не поступает других запросов, TCP-соединения обрываются.

Принцип работы веб-сервера

Когда пользователь вводит URL или щелкает на гиперссылке, браузер производит структурный анализ URL и интерпретирует часть, заключенную между *http://* и следующей косой чертой, как имя DNS, которое следует искать. Вооружившись IP-адресом сервера, браузер устанавливает TCP-соединение с 80 портом этого сервера. После этого отсылается команда, содержащая оставшуюся часть URL, в которой указывается путь к странице на

сервере. Сервер возвращает браузеру запрашиваемый файл для отображения.

В основном цикле сервер выполняет следующие действия:

1) принимает входящее TCP-соединение от клиента (браузера);

2) получает путь к странице, являющийся именем запрашиваемого файла;

3) получает файл (с диска);

4) высылает содержимое файла клиенту;

5) разрывает TCP-соединение.

Современные веб-серверы обладают более широкими возможностями, однако существенными в их работе являются именно перечисленные шаги, предпринимаемые в случае запроса контента, содержащегося в файле. В том случае, если контент является динамическим, третий шаг может быть заменен запуском программы (определенной по пути), возвращающей контент.

HTML — язык разметки веб-страниц

HTML (HyperText Markup Language) развивался вместе со Всемирной паутиной, разработка велась с 1986 г. по 1991 г. физиком Тимом Бернерсом-Ли. HTML является приложением к SGML (Standart Generalized Markup Language – стандартный обобщенный язык разметки), принятого Международной организацией по стандартизации (ISO). Язык HTML позволяет размещать на веб-страницах текст, изображения, видео, гиперссылки и т.п. Этот язык описывает способ форматирования документа, отделяя контент от его оформления.

Страница состоит из заголовка и тела. Вся страница размещается между командами форматирования, называемыми в языке HTML тегами <html> и </html>. Заголовок веб-страницы заключен в скобки тегов <head> и </head>, а тело располагается между тегами <body> и </body>. Команды внутри тегов называют директивами. Пусть не все, но большинство HTML-тегов имеют такой формат, то есть <tag> помечает начало чего-либо, а </tag> — его конец.

Регистр символов в тегах не имеет значения. Формат самого HTML-текста, то есть расположение строк и т. д., так же не имеет значения. Программы обработки HTML-текстов игнорируют лишние пробелы и переносы строк, так как они все равно форматируют текст так, чтобы он помещался в заданной области отображения. Соответственно, для того чтобы исходные HTML-документы легче читались, в них можно добавлять произвольное количество знаков табуляции, пробелов и символов переноса строк. И наоборот, для разделения абзацев в тексте в исходный HTML-текст недостаточно вставить пустую строку, так как она просто игнорируется браузером.

HTML 5 расширяет возможности HTML обладая возможностью обработки мультимедиа контента, популярного во Всемирной паутине. Браузер может проигрывать аудио и видео-контент, размещенный на веб-страницах, и для этого не требуется дополнительной установки плагинов. Поддержка векторной графики позволяет строить изображения в браузере, вместо использования растровых форматов графических объектов (таких как JPEG, PNG и GIF). Расширены возможности поддержки выполнения скриптов в браузерах, таких как фоновые потоки вычислений и доступ к хранилищу.

Таким образом веб-страницы становятся больше похожи на традиционные приложения с графическим пользовательским интерфейсом, чем на простые документы. Именно в этом направлении развивается Всемирная паутина.

CSS — каскадные таблицы стилей

Изначальной целью HTML было определение именно структуры документа, а вовсе не его внешнего вида [7]. Например *<h1>Заголовок</h1>* сообщает браузеру о том, что следует выделить заголовок, однако ничего не говорит о его гарнитуре, размере или цвете. Все эти детали реализует браузер по своему усмотрению: у него есть преимущество, состоящее в том, что он знает свойства конкретного монитора (например, сколько на нем точек). Но в какой-то момент разработчики веб-страниц захотели иметь контроль над

внешним представлением создаваемых страниц. Тогда были добавлены новые теги, отвечающие за внешний вид документа, и методы точного позиционирования элементов на экране. Но большая трудоемкость, которая сопутствовала данному подходу создания страниц, и плохое свойство переносимости ограничивали возможности контроля над внешним видом документа.

Альтернативным и гораздо более удобным способом является использование таблиц стилей. Таблицы стилей в текстовых редакторах позволяют авторам ассоциировать текст с логическим, а не с физическим стилем. Внешний вид каждого стиля определен отдельно.

CSS определяет простой язык для описания правил, которым подчиняется внешний вид размеченного тегами контента.

Таблицы стилей можно поместить в файл HTML (например, используя тег <*style*>), но обычно они хранятся в отдельном файле, на который дается ссылка.

У такой стратегии имеются два преимущества. Во-первых, она позволяет применить один набор стилей к нескольким страницам веб-сайта. Таким образом, мы получаем единообразный внешний вид страниц, даже если они разрабатывались разными авторами в разное время, кроме того, мы можем изменить внешний вид всего сайта, отредактировав только файл CSS, а не все файлы HTML.

Второе преимущество заключается в том, что загружаемые файлы HTML оказываются гораздо менее громоздкими, так как браузер может загрузить одну копию файла CSS для всех страниц, которые на нее ссылаются. Ему не нужно загружать новые копии всех определений для каждой отдельной веб-страницы.

Веб-приложения (динамические веб-страницы)

Статическая модель страниц, рассматривающая веб-страницы как документы связанные гипертекстовыми ссылками, соответствовала целям размещения информации онлайн в ранние дни развития Всемирной паутины. В

настоящее время Веб используется для приложений и сервисов, примерами которых являются покупка товаров в интернет-магазине, поиск по каталогам, использование картографических сервисов, чтение и отсылка почты, совместная работа с документами.

Эти новые типы использования похожи на традиционные приложения (например, программы для работы с почтой и текстовые редакторы). Отличие состоит в том, что эти приложения запускаются в браузере, а пользовательские данные хранятся на серверах в центрах обработки данных. Они используют веб-протоколы, получают информацию через Интернет, и браузер отображает пользовательский интерфейс. Пользователю не нужно устанавливать приложения, доступ к своим данным возможен с разных устройств и все данные сохраняются у оператора сервиса. Перечисленные преимущества дополняются тем, что крупные провайдеры предоставляют доступ к своим приложениям бесплатно. Эта модель является распространенной формой облачных вычислений, при которых вычисления перемещаются с пользовательских компьютеров на совместно используемые кластеры серверов в Интернете.

Веб-страницы больше не могут быть статичными, если они должны работать как приложения. Требуется динамический контент.

Простая ситуация, при которой необходима генерация страниц на стороне сервера, — это использование форм. Например, пользователь заполняет форму заказа и нажимает на кнопку Отправить заказ. При этом на сервер, на URL, определенный в форме, отсылается запрос, содержащий в себе данные, введенные пользователем. Эти данные должны быть переданы программе или скрипту для обработки. Таким образом, URL вызывает запуск определенной программы, в которую данные предоставляются в качестве входной информации. В этом случае обработка включает в себя введение заказа во внутреннюю систему, обновление записей клиента и списание денег с кредитной карты. Страница, которая возвращается в ответ на этот запрос, зависит от того, что произойдет в процессе обработки. Результат не фиксирован,

как в случае со статичными страницами. Если заказ успешно обрабатывается, возвращаемая страница может содержать дату доставки товара. Если запрос не был успешно обработан, возвращаемая страница может гласить, что запрашиваемых товаров нет в наличии или по какой-то причине не была принята кредитная карта.

То, как именно сервер запускает программу вместо поиска файла, зависит от устройства веб-сервера. Это не определяется самими веб-протоколами. Именно поэтому интерфейс может быть разработан в соответствии с требованиями компании-собственника сайта. Браузеру не нужно знать детали. Браузер просто создает запрос и получает страницу.

Тем не менее, для веб-серверов были разработаны стандартные API, чтобы запускать программы. Существование этих интерфейсов позволяет разработчикам тратить меньше усилий на расширение различных серверов за счет веб-приложений.

API является методом обработки запросов динамических страниц. Он был доступен с момента возникновения Всемирной паутины и называется CGI (Common Gateway Interface — общий шлюзовой интерфейс). CGI представляет собой интерфейс, позволяющий веб-серверам общаться с прикладными программами и скриптами, которые могут получать данные (например, из формы) и в ответ генерировать HTML-страницы. Эти программы могут быть написаны на любом выбранном разработчиком языке, обычно с использованием скриптов для простоты разработки.

Существует договоренность, в соответствии с которой программы, запускаемые через CGI, должны размещаться в каталоге CGI-BIN, который виден в URL. Сервер отображает запрос в этот каталог на имя программы и запускает программу как отдельный процесс. Он предоставляет программе любые данные, отосланные с запросом, как входные. На выходе программы получается веб-страница, передаваемая в браузер.

Второй API, серьезно отличается от уже описанного. Этот способ за-

ключается во внедрении небольших скриптов в HTML-страницы. Они выполняются на сервере, в их задачу входит генерирование страницы. Популярным инструментом для написания таких скриптов является PHP (Гипертекстовый препроцессор). При его использовании требуется, чтобы сервер понимал PHP. Обычно серверы определяют веб-страницы, написанные на PHP, по расширению *.php*, а не *.htm* или *.html*.

PHP использовать проще, чем CGI. Несмотря на простоту использования, PHP — это мощный язык программирования для взаимодействия со Всемирной паутиной и серверными базами данных. PHP имеет открытый исходный код, распространяется бесплатно и широко используется. PHP был разработан специально для сервера Apache, который также обладает открытым исходным кодом и является самым распространенным веб-сервером в мире.

Существуют другие методы генерации динамических страниц, перечислим некоторые из них.

JSP (JavaServer Pages — страницы сервера Java) в целом схож с PHP и отличается только тем, что динамическая часть программируется на языке Java. Файлы страниц, написанных с помощью JSP, имеют одноименное расширение: *.jsp*.

ASP. NET (Active Server Pages .NET — активные серверные страницы .NET) — это ответ Microsoft на PHP и JSP. Здесь для генерации динамического контента используются программы, написанные в собственной среде разработки сетевых приложений .NET, созданной Microsoft. Соответственно, файлы страниц, написанных с использованием этого метода, имеют расширение *.aspx*.

С точки зрения технологий все эти методы вполне сравнимы по возможностям.

Создание динамических веб-страниц на стороне клиента

Скрипты CGI и PHP решают вопросы обработки вводимых данных и

взаимодействия с базами данных, расположенными на сервере. Они могут принимать входящую информацию из форм, осуществлять поиск по одной или нескольким базам данных и в качестве результата генерировать HTML-страницы. Но ни один из этих методов не позволяет напрямую взаимодействовать с пользователем, например реагировать на движения мышкой. Для этих целей необходимы скрипты, внедренные в HTML- страницы и выполняющиеся не на серверной, а на клиентской машине. Начиная с HTML 4.0, появилась возможность включать скрипты такого типа с помощью тега *<script>*.

Наиболее популярный язык написания сценариев для клиентской стороны — это JavaScript. Как и другие языки написания скриптов, он очень высокоуровневый.

В случае с JavaScript браузер сам выполняет действия функции JavaScrit, содержащейся на странице. Вся работа производится локально, внутри браузера. С сервером никакого взаимодействия не осуществляется. Как следствие, результат появляется практически мгновенно, тогда как при использовании PHP задержка прибытия страницы с результатом может составлять несколько секунд.

Эти отличия вовсе не означают, что JavaScript лучше, чем например PHP. Просто у них различные сферы применения. PHP применяется тогда, когда необходимо взаимодействовать с удаленной базой данных. JavaScrit (и другие языки со стороны клиента используют тогда, когда требуется взаимодействие с пользователем в пределах его компьютера. Разумеется, возможно одновременное использование PHP и JavaScript.

AJAX — асинхронный JavaScript и XML

Сложным веб-приложениям требуются удобные пользовательские интерфейсы и простой доступ к данным, хранящимся на удаленных веб-серверах. Написание скриптов со стороны клиента (например, при помощи JavaScrit) и со стороны сервера (например, при помощи PHP) — это основ-

ные технологии, которые могут предоставить решение этих задач. Эти технологии обычно используются вместе с несколькими другими в комбинации под названием **AJAX** (Asynchronous JavaScript and XML — асинхронный JavaScript и XML).

AJAX это набор технологий, которые работают совместно, чтобы создать веб-приложения, которые были бы такими же удобными и функциональными, как традиционные настольные прикладные системы [8]. Перечислим и приведем краткую характеристику этих технологий:

– HTML и CSS, для представления информации в виде страниц;

– DOM (Document Object Model — объектная модель документов), для изменения части страниц при просмотре;

– XML (eXtensible Marku Lanuae — расширяемый язык разметки), для обмена данными приложений с сервером;

– асинхронный способ, при помощи которого программы отсылают и получают XML-данные;

– JavaScrit — язык, который связывает все эти технологии.

HTML и CSS были рассмотрены ранее.

DOM — это представление HTML-страницы, доступное программам. Это представление имеет структуру дерева, отражающего структуру HTML-элементов. Значение модели DOM состоит в том, что она предоставляет программам простой способ менять части страницы. Таким образом, не возникает необходимости полностью переписывать страницу, а меняется только тот узел, в который вносятся изменения. Когда они внесены, браузер обновит соответствующую часть страницы.

Третья технология, **XML** — это язык, предназначенный для описания структурированного контента. HTML мешает контент с форматированием, так как он связан со способом представления информации. Однако по мере того, как веб-приложения становятся более распространенными, нарастает потребность разделить структурированный контент и его представление.

В отличие от HTML для XML не существует четко определенных тегов. Каждый пользователь может создавать свои. XML хорошо подходит для передачи информации между программами, работающими в браузерах и на серверах, но ничего не говорит о том, каким образом документ должен быть представлен в виде веб-страницы. Язык под названием **XSLT** (eXtensible Stylesheet Language Transformations — стилевые трансформации расширяемого языка разметки) может использоваться для того, чтобы определить, каким образом трансформировать XML в HTML. XSLT похож на CSS, но у него гораздо больше возможностей.

Еще одно преимущество представления данных в виде XML, а не HTML состоит в том, что программам проще их анализировать. У XML гораздо более строгая структура. Имена тегов и атрибутов всегда пишутся в нижнем регистре, теги всегда должны закрываться в порядке, обратном порядку их появления, а значения атрибутов должны быть заключены в кавычки. Эта точность определений делает разбор проще, а результат получается однозначный.

Чтобы создать быстро откликающийся интерфейс в браузере при отсылке или получении данных, у скриптов должна быть возможность осуществлять асинхронные операции ввода/вывода, которые не блокируют дисплей при ожидании ответа на запрос.

JavaScript - скриптовый язык, который собирает AJAX в единое целое, предоставляя доступ к описанным выше технологиям. JavaScript был описан ранее.

На рисунке 1 изображена схема с обобщенными технологиями.

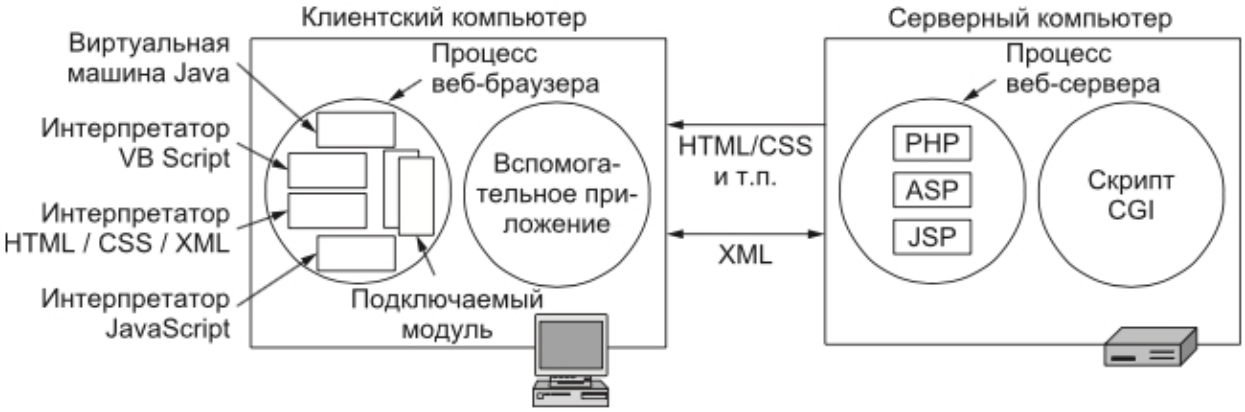

Рис.1. Различные технологии, используемые для создания динамических страниц

При помощи AJAX программы на веб-страницах могут асинхронно обмениваться XML-файлами и другими видами данных с сервером. Эта модель поддерживает различные веб-приложения, которые выглядят так же, как и традиционные приложения. Отличие лишь в том, что они работают в браузере и получают информацию, хранящуюся на серверах в Интернете.

Web 2.0

Термин Web 2.0 используется для обозначения новых тенденций в использовании технологии WWW, направленных на расширение творческих возможностей пользователей, более безопасный обмен информацией и взаимодействие между ними.

Термин получил распространение в 2004 году с подачи Тима О'Рейли для выражения нового способа взаимодействия разработчиков ПО и конечных пользователей. Ключевой принцип идеологии Web 2.0 был сформулирован как: «Интернет – как платформа» [9].

Web 2.0 можно рассматривать и как подход к построению систем, при котором они становятся тем лучше, чем больше людей ими пользуются в процессе сетевых взаимодействий. Таким образом, можно сказать, что Web 2.0 не является каким-либо четким стандартом, технологией, либо концепцией создания сайтов или их дизайна. Сейчас Web 2.0 – это скорее философия построения веб-приложений [10].

Фактически Web 2.0 означает переход веб-сайтов от изолированных накопителей информации к взаимосвязанным программным платформам,

воспринимаемых пользователями так, как будто они используются локально на его компьютере.

Ключевые технологии для Web 2.0:

- веб-сервисы – это программы, доступ к которым осуществляется через протокол HTTP, а обмен данными происходит в формате XML;
- использование технологий AJAX;
- одновременное распространение информации на различные веб-сайты;
- mash-up сервисы – сервис, который полностью или частично использует в качестве источников информации другие сервисы, предоставляя пользователю новую функциональность для работы.

В этом параграфе были рассмотрены основные технологии, на базе которых строят веб-приложения.

1.2. Теория разработки информационных систем

Под информационными ресурсами понимается информация, зафиксированная на материальном носителе и хранящаяся в информационных системах (библиотеках, архивах, фондах, банках данных и др.) [11].

Информационные ресурсы – это знания, подготовленные для целесообразного социального использования (документы, компьютерные базы данных, алгоритмы и программы автоматизированных устройств, произведения науки, литературы и искусства) [12].

Под информационной системой обычно подразумевается программная система, ориентированная на сбор, хранение, поиск и обработку информации. Подавляющее большинство информационных систем работает в режиме диалога с пользователем [13].

Процессы, обеспечивающие работу информационной системы любого назначения, условно можно представить состоящими из следующих блоков:

- ввод информации из внешних или внутренних источников;

- обработка входной информации и представление ее в удобном виде;
- вывод информации для представления потребителя или передача в другую систему;
- обратная связь – это информация, переработанная для корректной входной информации.

Собственно, информационный процесс – это «процесс создания, сбора, обработки, накопления, хранения, поиска, распространения и потребления информации» [14].

В зависимости от определенной области применения информационные системы могут очень сильно различаться своим функционалом, архитектуре и реализации. Но можно выделить, по крайней мере, два свойства, являющихся общими для всех информационных систем. Во-первых, любая информационная система предназначена для сбора, хранения и обработки информации. Поэтому в основе любой информационной системы лежит среда хранения и средства доступа к данным. Среда хранения должна обеспечивать надежный уровень хранения и эффективность доступа, в соответствии с областью применения информационной системы.

Во-вторых, информационные системы ориентируются на конечного пользователя. Такие пользователи могут быть очень далеки от мира компьютерных технологий, поэтому информационная система обязана обладать простым, удобным, понятным и легко осваиваемым интерфейсом, который должен предоставлять пользователю все необходимые для его работы функции.

Классификация информационных систем

Информационные системы классифицируются по разным признакам. Рассмотрим наиболее часто используемые способы классификации.

По масштабу информационные системы подразделяются на следующие группы (рис. 2) [15]:

- одиночные;

- групповые;
- корпоративные;

Рис. 2 Классификация информационных систем по масштабу

Одиночные информационные системы реализуются, как правило, на автономном персональном компьютере. Такая система может содержать несколько простых приложений, связанных общим информационным фондом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих по времени одно рабочее место. Такие приложения создаются с помощью так называемых настольных или локальных систем управления базами данных (СУБД).

Групповые информационные системы ориентированы на коллективное использование информации членами рабочей группы и чаще всего строятся на базе локальной вычислительной сети. При разработке таких приложений используются серверы баз данных (называемые также SQL-серверами) для рабочих групп.

Корпоративные информационные системы являются более масштабным вариантом систем для рабочих групп, они ориентированы на крупные компании и могут поддерживать территориально разнесенные узлы или сети. В основном они имеют иерархическую структуру из нескольких уровней. Для таких систем характерна архитектура клиент-сервер со специализацией серверов или же многоуровневая архитектура [16].

Для групповых и корпоративных систем существенно повышаются требования к надежности функционирования и сохранности данных. Эти свойства обеспечиваются поддержкой целостности данных, ссылок и тран-

закций в серверах баз данных.

По сфере применения информационные системы обычно подразделяются на четыре группы (рис. 3):

- системы обработки транзакций;
- системы принятия решений;
- информационно-справочные системы;
- офисные информационные системы.

Рис. 3 Классификация информационных систем по сфере применения

Системы обработки транзакций, по оперативности обработки данных так же разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций — OLTP (OnLine Transaction Processing), для отражения актуального состояния предметной области в любой момент времени.

Системы поддержки принятия решений — DSS (Decision Support System) — представляют собой другой тип информационных систем, в которых с помощью довольно сложных запросов производится отбор и анализ данных на различных уровнях: временных, географических и по другим по-

казателям.

Обширный класс информационно-справочных систем основан на гипертекстовых документах и мультимедиа. Наибольшее развитие такие информационные системы получили в сети Интернет.

Класс офисных информационных систем чаще служит для преобразования бумажных документов в электронный вид, автоматизацию делопроизводства и управление документооборотом.

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы (рис. 4):

- системы на основе архитектуры файл-сервера;
- системы на основе архитектуры клиент-сервера;
- системы на основе многоуровневой архитектуры;
- системы на основе Интернет/интранет-технологий.

Рис. 4 Классификация информационных систем по способу организации

Для создания удобных и простых в использовании и сопровождении информационных систем, эффективно работающих с базами данных, объединяют Интернет/интранет-технологии с многоуровневой архитектурой. При этом структура информационного приложения приобретает следующий вид: браузер — сервер приложений — сервер баз данных — сервер динамических страниц — веб-сервер.

Благодаря интеграции Интернет/интранет-технологий и архитектуры клиент-сервера, процесс внедрения и сопровождения корпоративной информационной системы заметно упрощается, сохраняя достаточно высокую эффективность и простоту совместного использования информации.

Требования, предъявляемые к информационным системам

Информационная система должна быть достаточно гибкой, надежной, эффективной и безопасной [17].

Гибкость, способность к адаптации и дальнейшему развитию, предполагает возможность приспособления информационной системы к новым условиям, новым потребностям компании. Выполнение этих условий возможно, если на этапе разработки информационной системы использовались общепринятые средства и методы документирования, так что даже спустя время будет возможно разобраться в структуре системы и внести в нее различные изменения, даже если все разработчики или их часть изменились или не смогут продолжить работу.

Надежность информационной системы подразумевает ее функционирование без искажения находящейся в ней информации и потери данных по «техническим причинам». Требование надежности обеспечивается созданием резервных копий хранимой информации, выполнением операций журнализации, поддержанием качества каналов связи и физических носителей информации, использованием современных программных и аппаратных средств. Сюда же следует отнести защиту от случайных потерь информации в силу недостаточной квалификации персонала.

Система будет *эффективной*, если с учетом выделенных ей ресурсов она позволяет решать возложенные на нее задачи в минимальные сроки. Эффективность системы обеспечивается оптимизацией данных и методов их обработки, применением оригинальных разработок, идей, методов проектирования.

Требование *безопасности* обеспечивается использованием новейших средств разработки информационных систем, методов защиты информации, применением современных аппаратных средств, паролирования и журнализации, постоянного мониторинга состояния безопасности операционных систем и средств их защиты.

Жизненный цикл информационных систем

В методологии проектирования информационных систем (ИС) под *жизненным циклом* ИС понимается непрерывный процесс, начиная с момента принятия решения о создании информационной системы и заканчивая моментом ее полного изъятия из эксплуатации [18]. Полный жизненный цикл ИС обычно включает в себя стратегическое планирование, анализ, проектирование, реализацию, внедрение и эксплуатацию.

Основной нормативный документ, регламентирующий состав процессов жизненного цикла – международный стандарт ISO/IEC 12207: 1995 «Information Technology – Software Life Cycle Processes» [19]. Он определяет структуру жизненного цикла, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания систем (его российский аналог ГОСТ ИСО/МЭК 12207-99 введен в действие в июле 2000 г.). В данном стандарте процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные.

В обсуждаемом стандарте все процессы жизненного цикла разделены на три группы:

- основные процессы, в состав которых входят процессы приобретения, поставки, разработки, эксплуатации, сопровождения;
- вспомогательные процессы – документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, совместная оценка, аудит, разрешение проблем;
- организационные процессы – управление, инфраструктура, усовершенствование, обучение.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами. Каждый процесс разделен на набор действий, каждое действие - на набор задач. Каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем не существует

заранее определенных последовательностей выполнения (естественно, при сохранении связей по входным данным). Стандарт ИСО 12207 пригоден для любых моделей жизненного цикла, методологий и технологий разработки ИС без конкретизации методов их реализации, действий и задач каждого из этапов жизненного цикла, но с описанием структуры этих процессов.

Так как модель жизненного цикла ИС зависит от условий, в которых она создается и функционирует, то необходимо учитывать предметную область, в которой будет работать будущая ИС. Кроме международного стандарта ISO/IEC 12207, существует корпоративный стандарт, разработанный компанией Rational Software, - один из лидеров на глобальном рынке программного обеспечения и средств для разработки ИС. Согласно этому стандарту жизненный цикл информационной системы включает четыре стадии:

1) начало – определение областей использования системы, а также выявление внешних для ИС объектов. Уточнение характера взаимодействия с внешними объектами на высоком уровне. Локализация возможностей системы с описанием наиболее существенных из них. Оценка критериев успеха разработки, уровня риска и объемов ресурсов, необходимых для выполнения разработки;

2) уточнение – анализ прикладной области и разработка архитектуры ИС с учетом специфики и назначения разрабатываемой ИС;

3) конструирование – разработка законченного изделия, готового к передаче пользователю. Оценка работоспособности разработанного программного обеспечения;

4) переход – передача разработанного ПО пользователям и его корректировка при обнаружении ошибок и недоработок. Определение степени достижения целей разработки.

Границы каждой стадии определяются временными отрезками, в которых следует принимать необходимые решения, приводящие к достижению главных целей проекта. Существенной особенностью жизненного цикла ИС

является цикличность: системный анализ - разработка - сопровождение - системный анализ. Это соответствует представлению об ИС как о развивающейся, динамической системе. На первом шаге процесса разработки системы создается проект ИС, а на последующих стадиях выполняется модификация проекта ИС, таким образом поддерживая его в актуальном состоянии. Со временем, модели жизненного цикла, которые определяют порядок выполнения стадий и этапов, претерпевали существенные изменения, наряду с технологиями проектирования ИС.

Модели жизненного цикла информационной системы

Модель жизненного цикла информационной системы – это структура, которая определяет порядок реализации процессов, действий и задач, выполняемых на протяжении жизненного цикла информационной системы, а также связи между процессами, действиями и задачами [20].

Стандарт ISO/IEC 12207 не определяет конкретные методы реализации, последовательность действий или выполнение задач, которые входят в процессы жизненного цикла информационной системы, а только описывает структурные сущности этих процессов. Это закономерно, так как стандарт ISO/IEC 12207 общий для всех моделей жизненного цикла, методологий проектирования и технологий разработки информационных систем. Специфика информационной системы, условия ее разработки и функционирования являются уникальными для каждой конкретной информационной системы и от совокупности этих факторов зависит модель жизненного цикла. Поэтому описание конкретных моделей жизненного цикла и методов разработки информационных систем для общего случая не имеет смысла, они не будут учитывать предметную область и совокупность факторов, которые оказывают влияние на процесс разработки и эксплуатации информационной системы.

В настоящее время широкое распространение получили каскадная и спиральная модель жизненного цикла информационной системы.

Каскадная модель жизненного цикла информационной системы

При использовании каскадной модели жизненного цикла весь процесс разработки информационной системы разбивается на этапы, работы на каждом этапе документируются. Переход с одного этапа на другой возможен только при полном завершении работ на предыдущем этапе. Последовательная организация работ является отличительной особенностью каскадной модели.

Перечислим основные этапы разработки, практически не зависящие от предметной области, рис. 5:

- анализ требований заказчика;
- проектирование информационной системы;
- разработка информационной системы;
- тестирование и опытная эксплуатация информационной системы;
- сдача готового программного продукта.

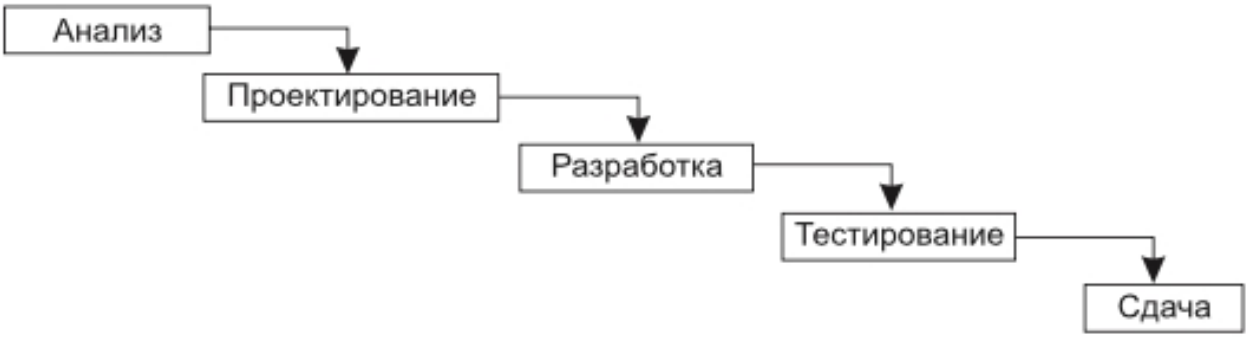

Рис. 5 Каскадная модель разработки

На первом этапе исследуются проблемы, которые должны быть решены, формулируются все требования заказчика. Результатом первого этапа, является техническое задание, которое согласуется со всеми заинтересованными сторонами.

На втором этапе осуществляется разработка проектных решений, которые должны соответствовать всем требованиям технического задания. Результатом второго этапа является набор проектной документации, которая содержит в себе все необходимые данные для реализации проекта.

Третий этап — реализация проекта. На этом этапе выполняется разра-

ботка программного обеспечения строго соответствующего проектным решениям, которые были сформулированы на предыдущем этапе. Методы реализации не имеют принципиального значения. В результате выполнения данного этапа появляется готовый программный продукт.

Четвертый этап - проверка программного обеспечения полученного на предыдущем этапе. Готовый продукт должен соответствовать всем требованиям, сформулированным в техническом задании. Пробная эксплуатация позволяет обнаружить недостатки работы информационной системы в условиях реальной работы.

Последним этапом является сдача готового проекта. На этом этапе необходимо продемонстрировать заказчику, что все его требования реализованы в полной мере.

Основные достоинства каскадной модели

Благодаря своим достоинствам каскадная модель хорошо проявила себя при выполнении инженерных разработок, вследствие чего получила широкое распространение. Рассмотрим основные достоинства каскадной модели:

- каждый этап сопровождается выпуском полного набора проектной документации. Пользовательская документация содержит всю необходимую информацию для поддержки информационной системы и разрабатывается на заключительных этапах жизненного цикла системы;
- возможность планировать сроки и затраты при последовательной организации процесса разработки.

Тем не менее, несмотря на свои достоинства, каскадная модель имеет ряд недостатков, ограничивающих ее применение при разработке информационных систем:

- существенная задержка получения результатов;
- ошибки и недоработки на любом из этапов выясняются, как правило, на последующих этапах работ, что приводит к необходимости возврата

на предыдущие стадии;

- сложность распараллеливания работ по проекту;
- чрезмерная информационная перенасыщенность каждого из этапов;
- сложность управления проектом;
- высокий уровень риска и ненадежность инвестиций.

Задержка получения результатов считается главным недостатком каскадной модели. Этот недостаток появляется потому, что при последовательном подходе к разработке согласование результатов с заказчиком осуществляется после того, как завершен очередной этап работ. По этой причине может случиться так, что разрабатываемая информационная система не соответствует требованиям пользователей. И такие проблемы могут возникнуть на любом из этапов разработки. Проектировщики-аналитики и программисты могут непреднамеренно искажать систему, так как они не всегда хорошо разбираются в областях, для которых производится разработка информационной системы.

Возврат на предыдущие стадии. Этот недостаток является следствием предыдущего. Ошибки, допущенные на более ранних этапах, часто, обнаруживаются только на более поздних стадиях работы над проектом. После того, как ошибки проявятся, проект необходимо вернуть на предыдущий этап, переработать, исправить ошибки и снова вернуть на последующую стадию. Это является причиной срыва графика работ и может привести к обострению отношений между разработчиками, которые выполняют отдельные этапы работ.

Сложность параллельного ведения работ. Вследствие того, что этапы выполняются последовательно, сложно организовать параллельную работу разработчиков, в результате возможны простои, когда одна группа ждет результат работы другой.

Информационная перенасыщенность. Причиной служит сильная зави-

симость между различными группами разработчиков. Данная проблема проявляется в том, что при изменении одной из частей проекта необходимо уведомить всех разработчиков, которые могли использовать эту часть в своей работе. Когда система состоит из большого количества взаимосвязанных подсистем, появляется важная самостоятельная задача синхронизации актуальной документации.

Строгая последовательность разработки системы и наличие сложных взаимосвязей между различными частями усложняет управления проектом при использовании каскадной модели. Последовательная разработка проекта приводит к тому, что одни группы разработчиков должны ожидать результатов работы других команд. В результате требуется административное вмешательство, чтобы согласовать сроки выполнения работ и состава документации, передаваемой заказчику.

Высокий уровень риска. Чем сложнее проект, тем дольше продолжительность каждого из этапов разработки и тем сложнее взаимосвязи между отдельными частями проекта. А результаты разработки можно увидеть и оценить только лишь на этапе тестирования, когда завершен анализ, проектирование и разработка системы, которые требуют значительного времени и средств.

Сложные проекты, разрабатываемые по каскадной схеме, имеют повышенный уровень риска. Этот вывод подтверждается практикой: по сведениям консалтинговой компании The Standish Group, в США более 31 % проектов информационных систем заканчивается неуспехом; почти 53 % IT-проектов завершается с перерасходом бюджета (в среднем на 189 %, то есть почти в два раза); и только 16,2 % проектов укладывается и в срок и в бюджет [21].

Спиральная модель жизненного цикла

Разработка информационной системы при использовании спиральной модели жизненного цикла ведется итерациями. Итерационный подход к раз-

работке подразумевает циклическую организацию процесса. При этом самыми важными являются начальные этапы анализа и проектирования системы. На этапе анализа проверяется возможность реализации и жизнеспособность системы, а на этапе проектирования создается прототип будущей ИС.

В начале работы над проектом у заказчика отсутствует четкое видение финального продукта, и требования к информационной системе не могут быть четко определены. Возможно, отсутствует уверенность в успешной реализации проекта, поскольку риски очень велики. Поэтому разработка системы ведется по частям с возможностью изменений требований или отказа от ее дальнейшего развития. Развитие проекта может быть завершено не только после стадии внедрения, но и после стадии анализа рисков.

Каждая итерация характеризуется выпуском версии продукта, являющейся частью законченной системы, и с каждым новым витком разработки продукт эволюционирует, рис. 6. Помимо выпуска продукта на каждом витке спирали уточняются цели проекта и характеристики продукта, которые необходимо достичь, определяется качество реализованного решения и план работы над следующей итерацией. На каждой итерации проект становится более детализированным, по мере развития в него вносятся коррективы с учетом обновленных требований к конечному решению и проект доводится до финальной реализации.

Согласно спиральной модели переход на следующий этап разработки возможен вне зависимости от того полностью завершена работа на предыдущем этапе или нет. Все незавершенные работы возможно реализовать на следующей итерации. Эта особенность спиральной модели позволяет снизить уровень рисков, не позволяя затягивать время разработки информационной системы.

Рис. 6 Спиральная модель жизненного цикла информационной системы

Наличие на ранней стадии работоспособного продукта, который можно показать пользователям, упрощает процесс доработки и внесения уточнений в систему, и чем раньше вскроются недостатки и несоответствия, тем проще вносить эти изменения. Поэтому основной задачей каждой итерации является быстрое создание работающего решения.

Преимущества спиральной модели

- позволяет изменить требования заказчика, что характерно для большинства разработок;
- при использовании спиральной модели отдельные элементы информационной системы интегрируются в единое целое постепенно. При итерационном подходе интеграция производится фактически непрерывно. Поскольку интеграция начинается с меньшего количества элементов, то возникает гораздо меньше проблем при ее проведении;
- уменьшение уровня рисков. Уровень рисков максимален в начале разработки проекта. По мере продвижения разработки ожидаемый риск

уменьшается. Так же возможен отказ от проекта с минимальными финансовыми затратами;

- обеспечивает большую гибкость в управлении проектом;
- итерационный подход упрощает повторное использование компонентов;
- спиральная модель позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места выявляются и исправляются на каждой итерации;
- совершенствование процесса разработки — анализ, проводимый в каждой итерации, позволяет оценить организацию процесса разработки и внести улучшения на следующей итерации.

Основная проблема спирального цикла — определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход должен осуществляться в соответствии с планом, даже если не вся работа выполнена. При итерационном подходе полезно следовать принципу «лучшее — враг хорошего». Поэтому завершение итерации должно производиться строго в соответствии с планом [22].

Методологии проектирования информационных систем

Методология создания информационных систем заключается в организации процесса их построения и обеспечения управления этим процессом для того, что бы гарантировать выполнение требований как к самой системе, так и к характеристикам процесса разработки.

Основными задачами, решение которых должна обеспечивать методология создания информационных систем (с помощью набора инструментальных средств), являются следующие:

- обеспечение создания ИС, отвечающим целям и задачам и предъявляемым к ним требованиям;
- гарантия создания системы с заданными параметрами в течении

заданного времени в рамках бюджета;

- простота сопровождения, модификации и расширения системы с целью обеспечения ее соответствия изменяющимся условиям работы.

Методологии, технологии и инструментальные средства проектирования составляют основу проекта любой информационной системы. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов жизненного цикла информационных систем.

Основное содержание технологии проектирования составляют технологические инструкции, состоящие их описания последовательности технологических операций, условий, в зависимости от которых выполняется та или иная операция.

Для успешной реализации проекта объект проектирования (ИС) должен быть, прежде всего, адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС.

Для разработки корпоративной информационной системы для записи детей в детские сады будем использовать технологию AJAX для создания веб-приложения с удобным пользовательским интерфейсом и простым доступом к данным, хранящихся на удаленных веб-серверах. Процесс разработки проекта строится на основе каскадной модели жизненного цикла информационной системы с разбиением всего процесса на этапы и последовательной организацией работ.

2. ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ КОРПОРАТИВНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ЗАПИСИ ДЕТЕЙ В ДЕТСКИЕ САДЫ

2.1. Анализ актуальности и существующих решений

Реализация принятых в рамках Концепции демографической политики Российской Федерации на период до 2025 года мер, направленных на стимулирование рождаемости, таких, как введение ежемесячного пособия по уходу за ребенком неработающим женщинам, увеличение размера пособия по беременности и родам и ежемесячного пособия по уходу за ребенком работающим женщинам, введение родового сертификата и налоговые льготы оказали позитивное влияние на демографическое положение страны.

По данным Росстата, с начала 2000-х годов число родившихся непрерывно растёт, а с середины 2000-х число умерших непрерывно падает.

В 2014 году впервые в современной истории России в стране родилось 1,947 млн. детей [23].

К концу 2014 года в России смертность среди детей до пяти лет снизилась в три раза по сравнению с 1990 годом.

Рождаемость в мае 2016 года выросла по сравнению с аналогичным периодом прошлого года на 5 %. Всего за 5 месяцев (с января по май) родилось 762,5 тысяч детей, что на 1,5 тысячи детей больше, чем за тот же период 2015 года.

Текущее состояние дел лучше всего показывает суммарный коэффициент рождаемости, рис.7. Он не зависит от возрастной структуры населения, так как вычисляется путём сложения уровней рождаемости всех возрастов на выбранный год.

Суммарный коэффициент рождаемости в России практически непрерывно растёт с 1999 года. Тогда он составлял 1,16 детей на одну женщину, а в 2016 году вырос до более 1,8 [24]. По этому показателю Россия входит в

первую десятку европейских стран с самым высоким уровнем рождаемости и занимает первое место среди европейских стран по темпам ежегодного прироста уровня рождаемости за последние 5 лет [25].

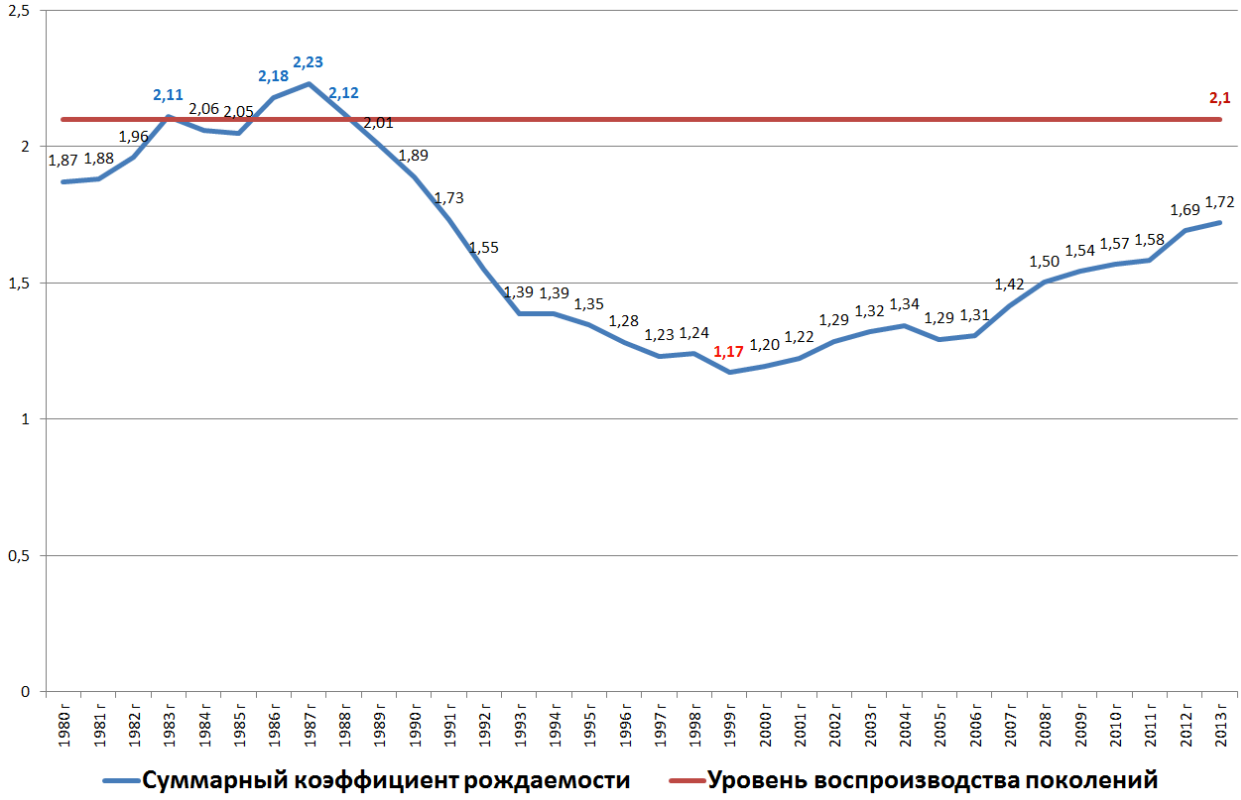

Рис.7. Суммарный коэффициент рождаемости в России в 1980–2013 годах

Вследствие роста рождаемости и того, что в 1990-ые годы большое количество детский учреждений было перепрофилировано в России обострилась проблема дефицита мест в детских садах.

Истоки проблемы нехватки детских садов связаны с падением рождаемости в 90-е годы. Недостаточная загруженность детских садов привела к их массовым закрытиям государством, переоборудованию, а также к передаче в частные руки. Сокращение числа детских садов было естественно с точки зрения реальной политики, но не учитывало долговременные государственные интересы. Изменение в демографической ситуации привело к потребности в большем количестве детских садов. Повышение рождаемости неизбежно привело к массовым очередям, переуплотнениям в детских садах. Такая картина характерна для многих городов России.

Альтернативой муниципальным детским садам являются частные дет-

ские сады. В городе Екатеринбург насчитывается более 250 частных детских садов и их количество постоянно увеличивается [26]. В связи с этим, становятся актуальны и интернет-сервисы предоставляющие информацию о данных учреждениях.

Рассмотрим наиболее популярные сервисы, предоставляющие информацию об учреждениях, в городе Екатеринбурге.

Первый сервис – **«Каталог частных детских садов»** доступен по адресу http://det-sad.net/, рис. 8.

Данный сервис предоставляет информацию по многим городам России, в том числе и в г. Екатеринбург. Сервис располагает обширным каталогом частных детских садов.

Основная идея сервиса – предоставление информации об учреждениях.

В каталоге присутствует возможность отбора учреждений по районам города, рис. 9, а также интерактивная карта со всеми учреждениями.

На странице детского сада представлена следующая информация, рис.10:

- название учреждения;
- адрес (в том числе на интерактивной карте);
- телефон;
- официальный сайт учреждения;
- режим работы;
- ежемесячная плата;
- возраст детей.
- присутствует возможность оставить комментарий через сторонний сервис «Cacle».

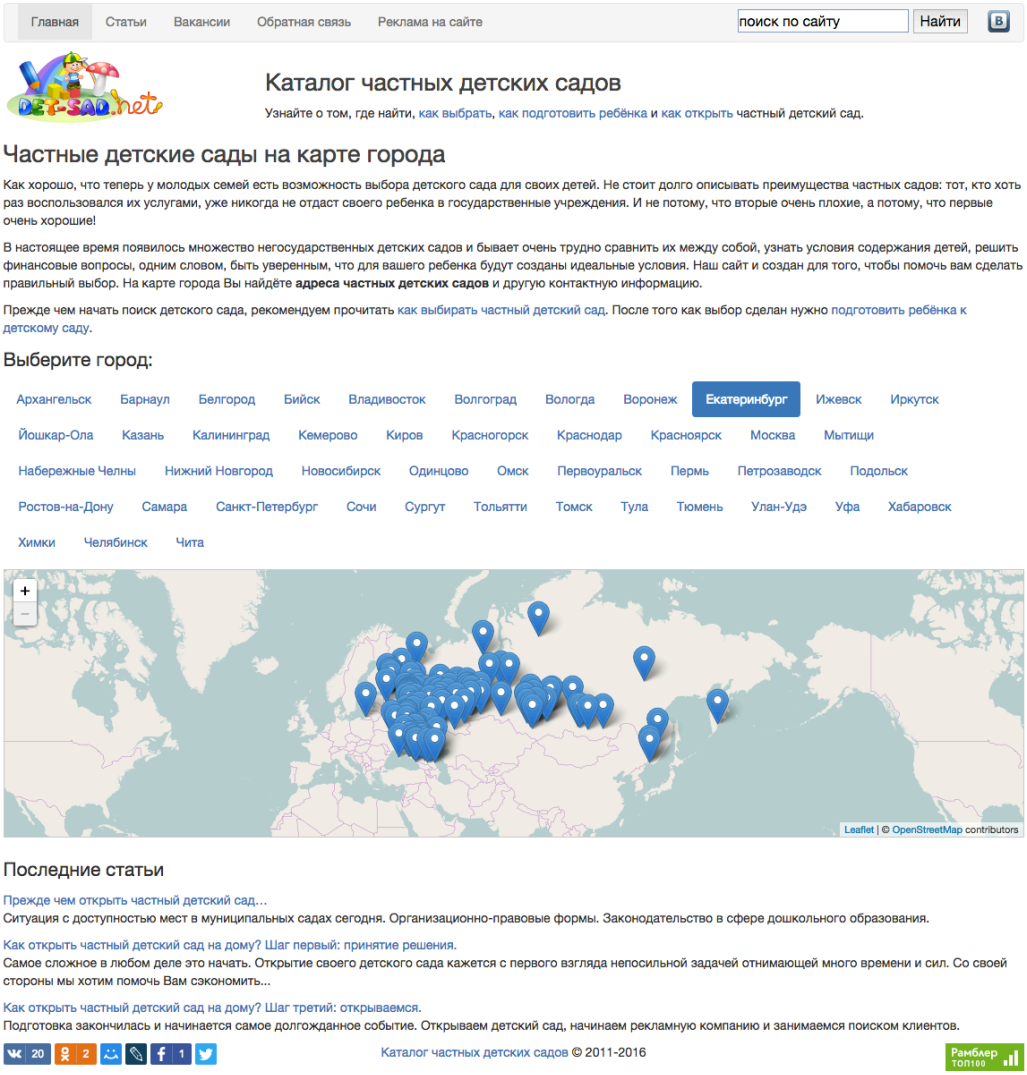

Рис.8. Интернет-сервис «Каталог частных детских садов»

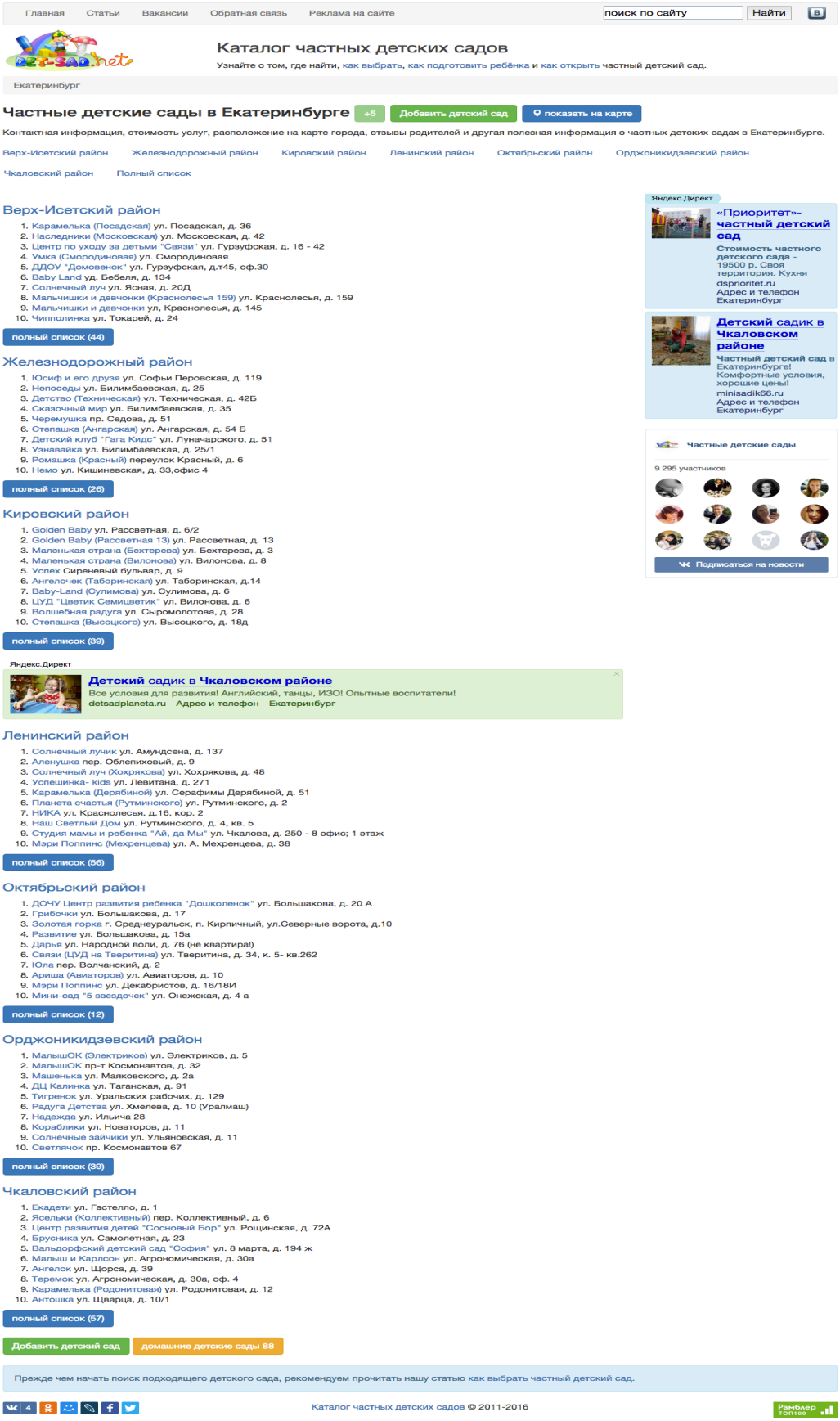

Рис.9. Каталог детских садов, с отбором учреждений по районам города

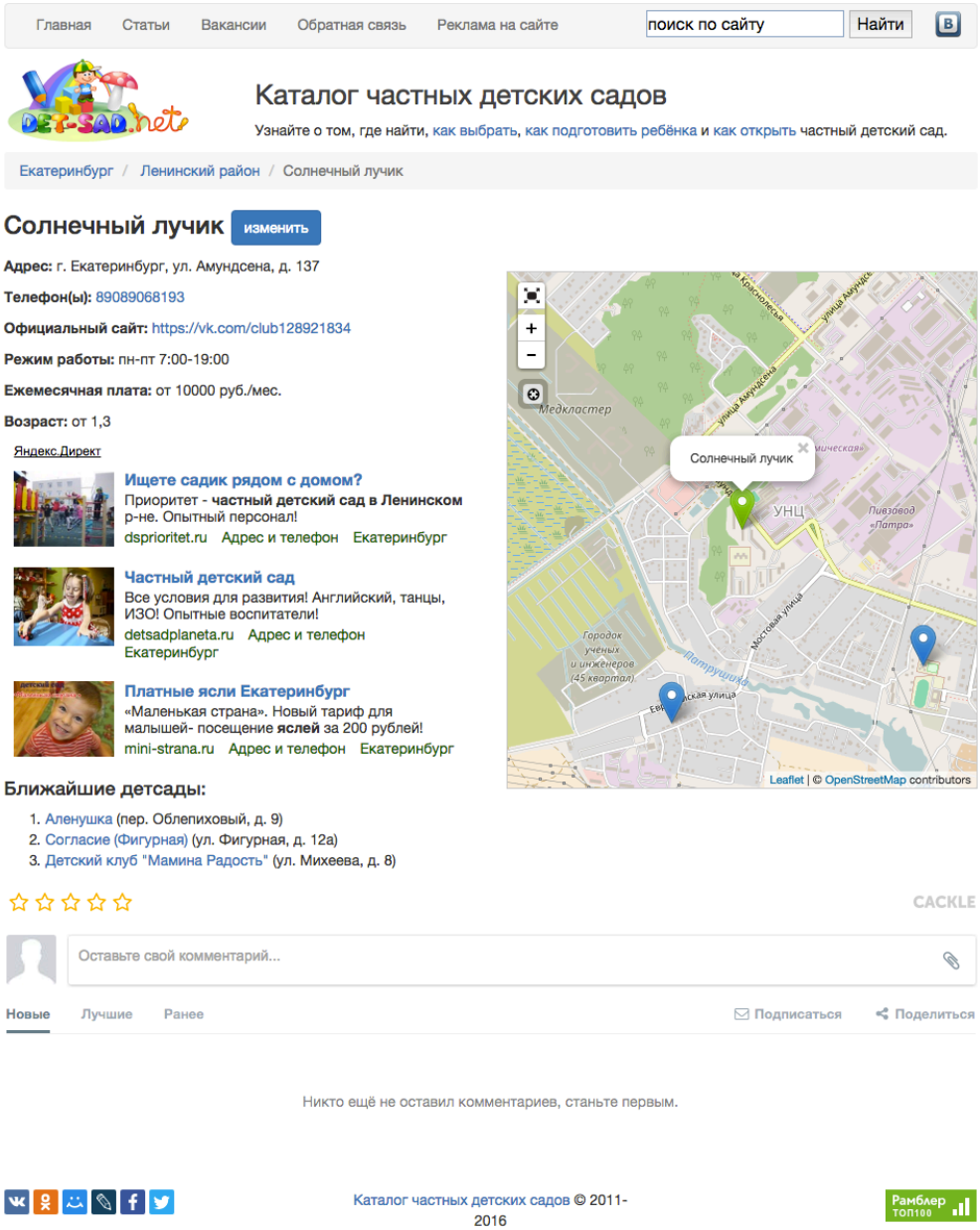

Рис.10. Страница детского сада

Анализ сайта позволяет выделить основные преимущества сервиса «Каталог частных детских садов»:

- большая база данных учреждений;
- интерактивная карта с каталогом учреждений;
- присутствует фильтрация по районам города;
- простая и понятная навигация по сервису.

Также были выявлены следующие недостатки сервиса:

- не предоставляется информация о наличии или отсутствии мест в детских садах;

- отсутствует функционал записи в детские сады или подачи заявки на запись;
- на странице детского сада отсутствует описание учреждения, дается только контактная информация;
- устаревший дизайн сервиса.

Сервис «Каталог частных детских садов» является неплохим информационным ресурсом, с большой базой учреждений и простой навигацией по сайту. Но отсутствие важного функционала, каким является возможность подачи заявки на запись непосредственно с сайта сервиса и отсутствие информации о доступности мест, делает сервис ограниченным в использовании и не предоставляет тех услуг, которые мог бы предоставить.

Второй сервис, предоставляющий услуги по поиску детских садов – это **доска объявлений на сайте портала E1.ru** - http://do.e1.ru/categories/services/nursery/quotient/, рис. 11.

Данный сервис является доской объявлений, на которой объявления сортируются по дате и периодически удаляются, согласно условиям сервиса. Вследствие этого, в актуальном состоянии находится ограниченный набор учреждений и база не является полной. Пользователям доступна сортировка по дате и стоимости. Система фильтров включает в себя отбор по районам города, цене и другим дополнительным параметрам.

Основные преимущества данного сервиса:

- хорошая навигация;
- возможность гибкого поиска с учетом стоимости и района, доступна сортировка по цене и дате;
- современный дизайн сервиса.

Недостатки сервиса:

- не полная база учреждений;
- не предоставляется информация о наличии или отсутствии мест в детских садах;

- отсутствует функционал записи в детские сады или подачи заявки на запись;
- отсутствует интерактивная карта учреждений;
- отсутствует сортировка по возрасту;
- отсутствие дополнительной полезной информации по вопросу выбора детского сада.

Так как сервис не является специализированным, он не способен предоставить исчерпывающую информацию для выбора детского сада. Пользователь получает довольно ограниченную, хоть и актуальную информацию.

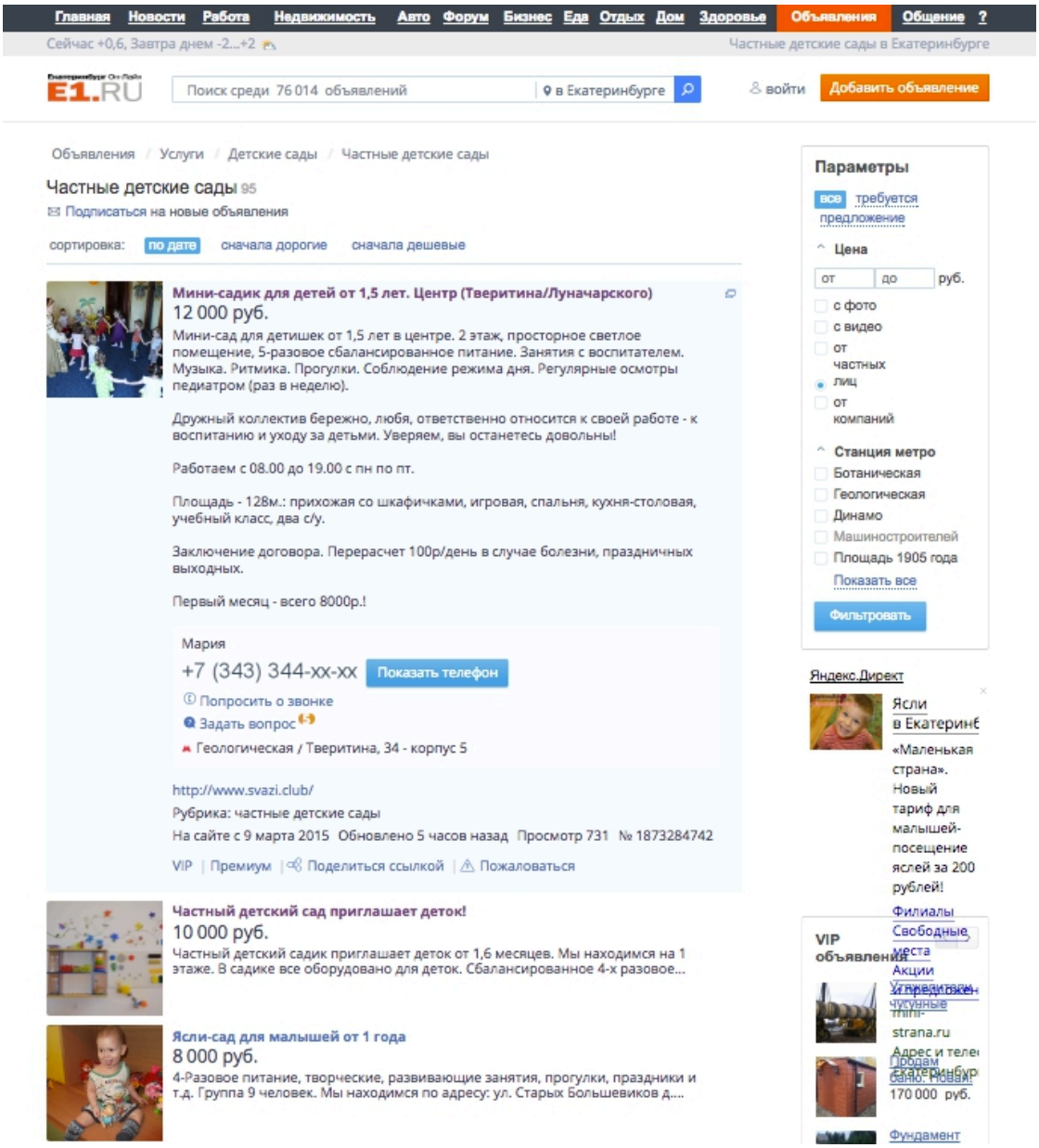

Рис. 11. Объявления на сайте E1.ru

Третий сервис – **«Детские сады и мини-садики Екатеринбурга»** доступен по адресу http://detsad-ural.ru/, рис. 12.

Данный ресурс представляет из себя каталог детских учреждений в городе Екатеринбург.

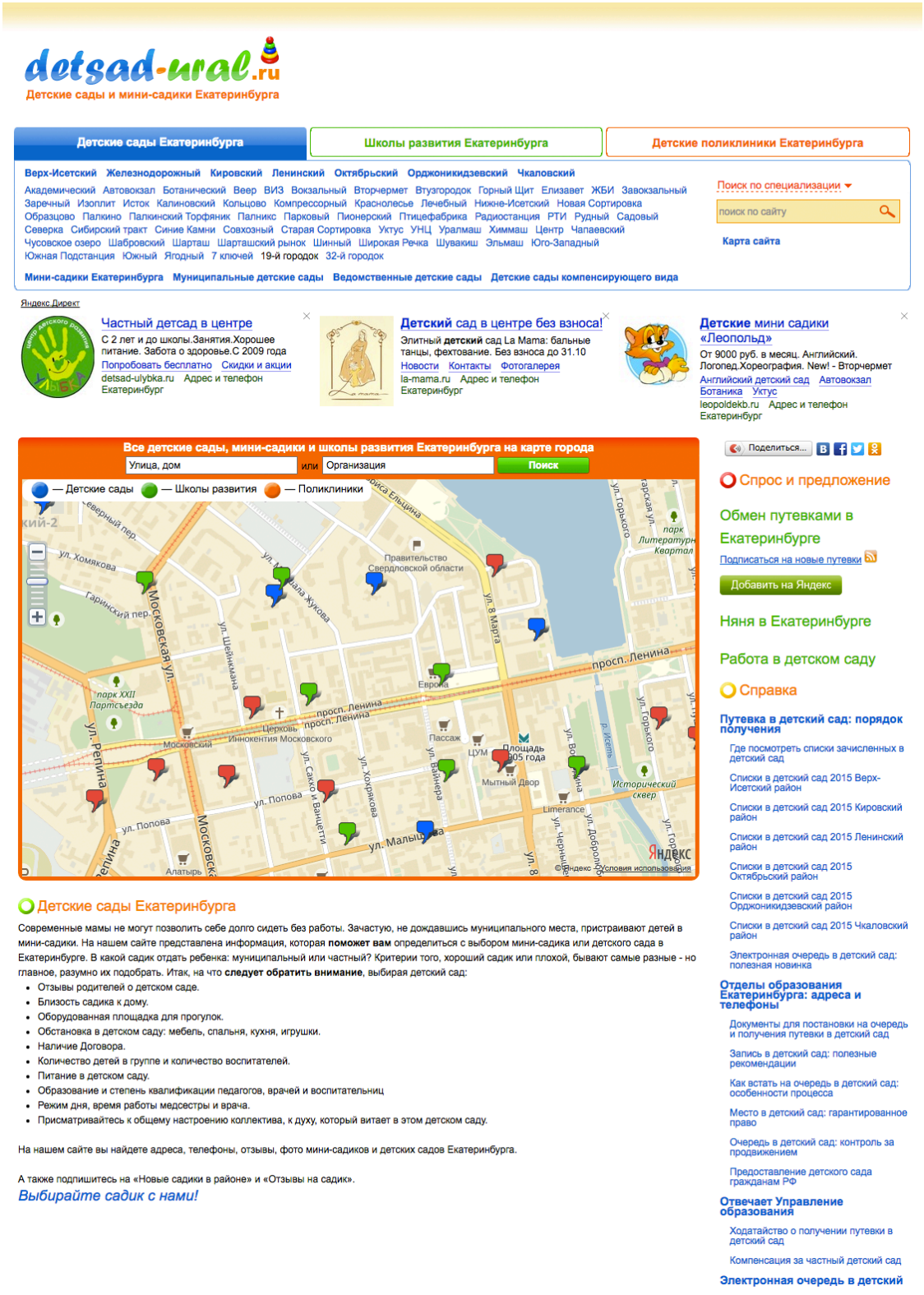

Рис. 12. Главная страница сервиса «detsad-ural.ru»

Укажем основные преимущества данного сервиса:

- наличие интерактивной карты с учреждениями;
- отбор по районам города.

Анализ сайта позволил выявить следующие недостатки сервиса:

- маленькая база учреждений;
- не предоставляется информация о наличии или отсутствии мест в детских садах;
- отсутствует функционал записи в детские сады или подачи заявки на запись;
- отсутствует система фильтров для гибкого отбора учреждений;
- отсутствует сортировка учреждений в результатах поиска;
- неудобная навигация по сайту. Для того, что бы разобраться в большой ссылочной массе и выделить необходимые разделы, необходимо потратить значительное количество времени и когнитивных способностей;
- отсутствует дизайн веб-сервиса.

Использование данного сервиса позволяет сделать вывод о том, что ресурс проектировался очень давно и сильно устарел. Отсутствие простой и ясной навигации, а также важного функционала делает использование сервиса крайне утомительным и не продуктивным.

Проведя анализ наиболее популярных сервисов, связанных с частными детскими садами, можно сделать вывод о том, что ни один из сервисов не предоставляет такой важной услуги, как возможность записаться в учреждение непосредственно с сайта. В целом, все сервисы выглядят устаревшими и не соответствующими современным представлениям, и существует необходимость в разработке и создании современного сервиса для записи детей в детские сады.

2.2. Разработка корпоративной информационной системы

Название «Cheep», с английского языка переводится как «писк», от-

лично подходит для нашего сервиса, т.к. связано с детской тематикой, приятно звучит и легко запоминается. Сервис доступен по адресу http://cheeping.ru/.

Цель создания сервиса

Основной целью создания сервиса является предоставление услуг по записи ребенка в детские сады, без необходимости траты времени на длительный поиск, посещение и обзвон учреждений. Пользователи получают актуальную информацию и непосредственно с сайта осуществляют запись ребенка в учреждение. Таким образом, пользователи находят то учреждение, которое им подходит.

Определение целевой аудитории пользователей сайта

Проанализировав демографические данные из открытых источников и проведя интервью с заинтересованными лицами, составлены портреты людей, которые нуждаются в услугах данного сервиса.

Портрет 1-го типа пользователей:

- пол: женский;
- возраст: 20 – 38 лет. По данным ресурса «Акушерство сегодня» средний возраст рожениц составляет 21-26 лет [27];
- роль: мать, ухаживающая за ребенком. Нуждается в услугах детских садов;
- уровень владения ПК: активный пользователь персонального компьютера, в основном для общения в социальных сетях и просмотра медиа-контента ;
- обучаемость: низкая, не заинтересован в освоении сложных интерфейсов;
- контекст использования: дома, имеет отдельный стол с ПК;
- цель: устроить ребенка в детский сад;
- задачи: поиск учреждения, соответствующего своим потребностям и запись ребенка в детский сад.

Портрет 2-го типа пользователей:

- пол: мужской;
- возраст: 20 – 42 года;
- роль: отец, занимающийся поиском детского сада для своего ребенка;
- уровень владения ПК: активный пользователь персонального компьютера, в основном для общения в социальных сетях и просмотра медиа-контента ;
- обучаемость: низкая, не заинтересован в освоении сложных интерфейсов;
- контекст использования: дома, имеет отдельный стол с ПК;
- цель: устроить ребенка в детский сад;
- задачи: поиск учреждения, соответствующего своим потребностям и запись ребенка в детский сад.

Сценарий действий для обоих пользователей похож, поэтому распишем сценарий для одного пользователя. Женщине 27 лет - Ирине Николаевне, два года назад родившей ребенка, необходимо пристроить сына в детский сад, для того, что бы продолжить свою трудовую деятельность. Зная о преимуществах частных детских садов и нехватки мест в муниципальных учреждениях, решает отдать сына именно в частный детский сад. Поэтому Ирина Николаевна открывает браузер, заходит на страницу поисковика «Яндекс» и начинает искать по запросу «Частные детские сады Екатеринбург». Просматривая поисковую выдачу, решает перейти на страницу с названием «Запись в частные детские сады». Тем самым попадает на главную страницу сайта сервиса «Cheep». Ей требуется найти частный детский сад. Основные условия поиска – близость к дому и доступная цена. После отбора детских садов из каталога сервиса, чтения их описания, Ирина Николаевна решает остановить свой выбор на одном учреждении и записать сына именно в него. Просматривая сайт, Ирина Николаевна видит, что сервис предоставляет возможность запи-

сать ребенка в детский сад онлайн и женщина решает воспользоваться данной возможностью. Введя необходимые данные в форму записи и отправив ее, женщина подала заявку на запись своего ребенка в учреждение.

Разработка интерфейса

На основе сценария и фактах о пользователях начинаем создавать прототип веб-интерфейса.

Прототипирование программного обеспечения — это этап разработки программного обеспечения, процесс создания прототипа программы — макета (черновой, пробной версии) программы, обычно — с целью проверки пригодности предлагаемых для применения концепций, архитектурных и/или технологических решений, а также для представления программы заказчику на ранних стадиях процесса разработки.

Прототип позволяет также получить обратную связь от будущих пользователей, причем, именно тогда, когда это наиболее необходимо: в начале проекта еще есть возможность исправить ошибки проектирования практически без потерь [28].

Существует огромное количество инструментов для прототипирования, наиболее популярными являются Axure RP, Balsamiq Mockups, Moqups, Wireframe.cc и OmniGraffle. Всех их объединяет схожий по своей сути функционал, но в отдельных программах лучше реализована интерфейсная часть, в других более обширна библиотека шаблонов и присутствуют/отсутствуют дополнительные функции командной работы над проектом, создания интерактивных прототипов и симуляции основных пользовательских действий.

Для прототипирования сервиса нами выбрана программа Balsamiq Mockups. Он позволяет быстро сделать набросок будущего интерфейса, будь то веб-сайт или окно клиентского приложения, выбрав из более 60 различных готовых объектов. Основное его преимущество состоит в том, что выполненный в нем прототип выглядит как «бумажный» рисунок и лишь отдаленно напоминает готовый продукт. Это полезно тем, что концентрирует внимание

разработчика на общей концепции дизайна, расположении элементов, функциональности и наполнении форм, и не дает зациклиться на том, как сделать «красивее», и уйти от основной задачи. Интерфейс его легок в освоении и интуитивно понятен. Кроме этого позволяет сохранять свой прототип в PNG или PDF формате для последующей демонстрации на любом устройстве.

Определение структурных элементов

На основе концепции сервиса и анализа сайтов конкурентов была составлена структурная схема, рис. 13.

- главная страница;
- каталог с системой фильтров;
- страница учреждения с формой записи;
- страница «Добавить учреждение»;
- страница «О нас»;
- интерактивная карта с каталогом учреждений;
- страница «Рекламодателям»;
- страница «Контакты»;
- текстовая страница.

Главная страница содержит промо-блок, поиск с системой фильтров, рекламный блок и блок статей. Каталог содержит систему фильтров, вывод отобранных результатов, результаты могут сортироваться по стоимости. Страница учреждения содержит описание конкретного учреждения и форму записи в это учреждение.

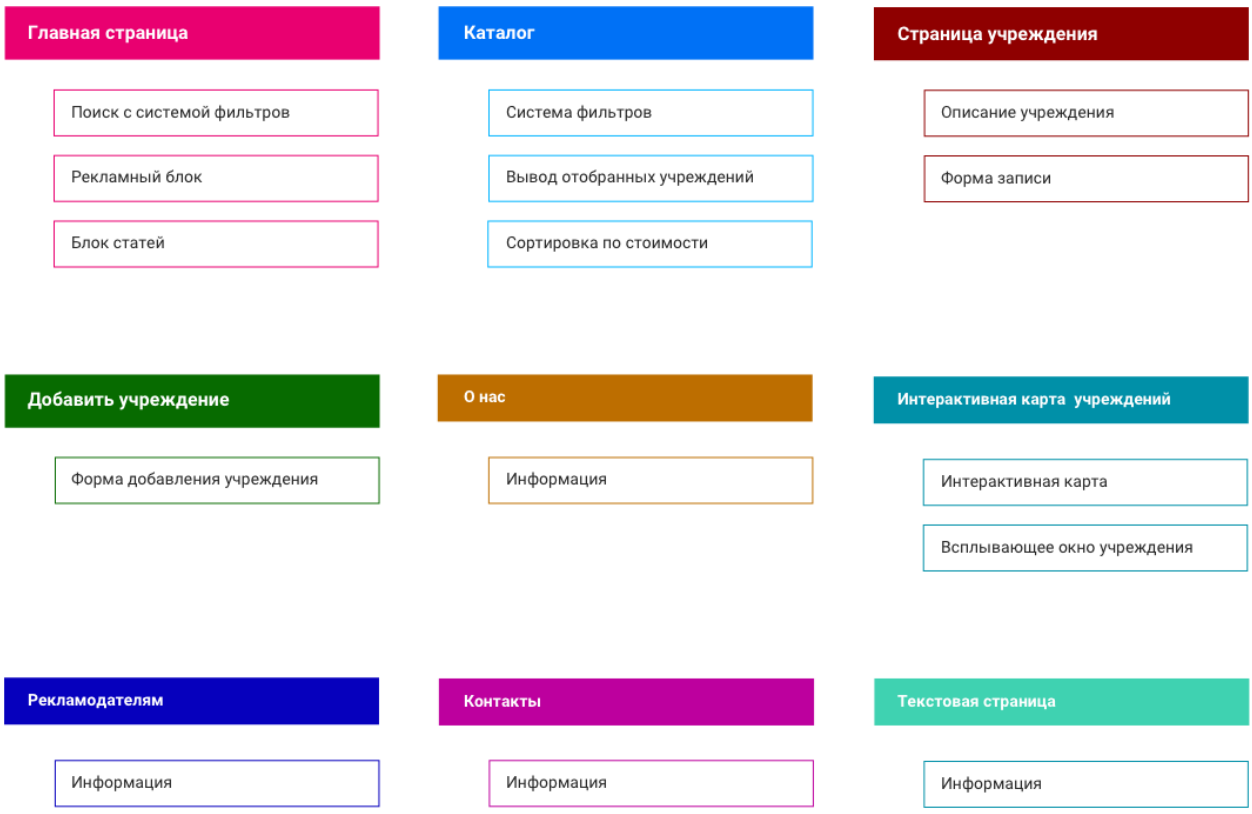

Рис. 13. Структурная схема сервиса

Этап прототипирования

В программе Balsamiq Mockups были созданы макеты интерфейсов будущего сервиса. Прототип главной страницы можно увидеть на рис. 14.

Рассмотрим прототип более подробно, разделив на смысловые блоки.

В «шапке» располагаются логотип компании и средства навигации по сайту. Расположение этих элементов в шапке сайта является предпочтительным для такого типа сайта, т.к. для пользователя расположение этих элементов именно вверху страницы является привычным в среде веб, что снижает когнитивную нагрузку на пользователя, позволяя быстрее освоиться на сайте и искать нужную ему информацию вместо того, что бы прилагать усилия к освоению нового интерфейса.

Промо-блок содержит изображение, промо-текст и подсказку по работе с сервисом в виде шагов, которые необходимо выполнить для достижения цели. Кнопка «Начать поиск» проматывает страницу до поисковой формы.

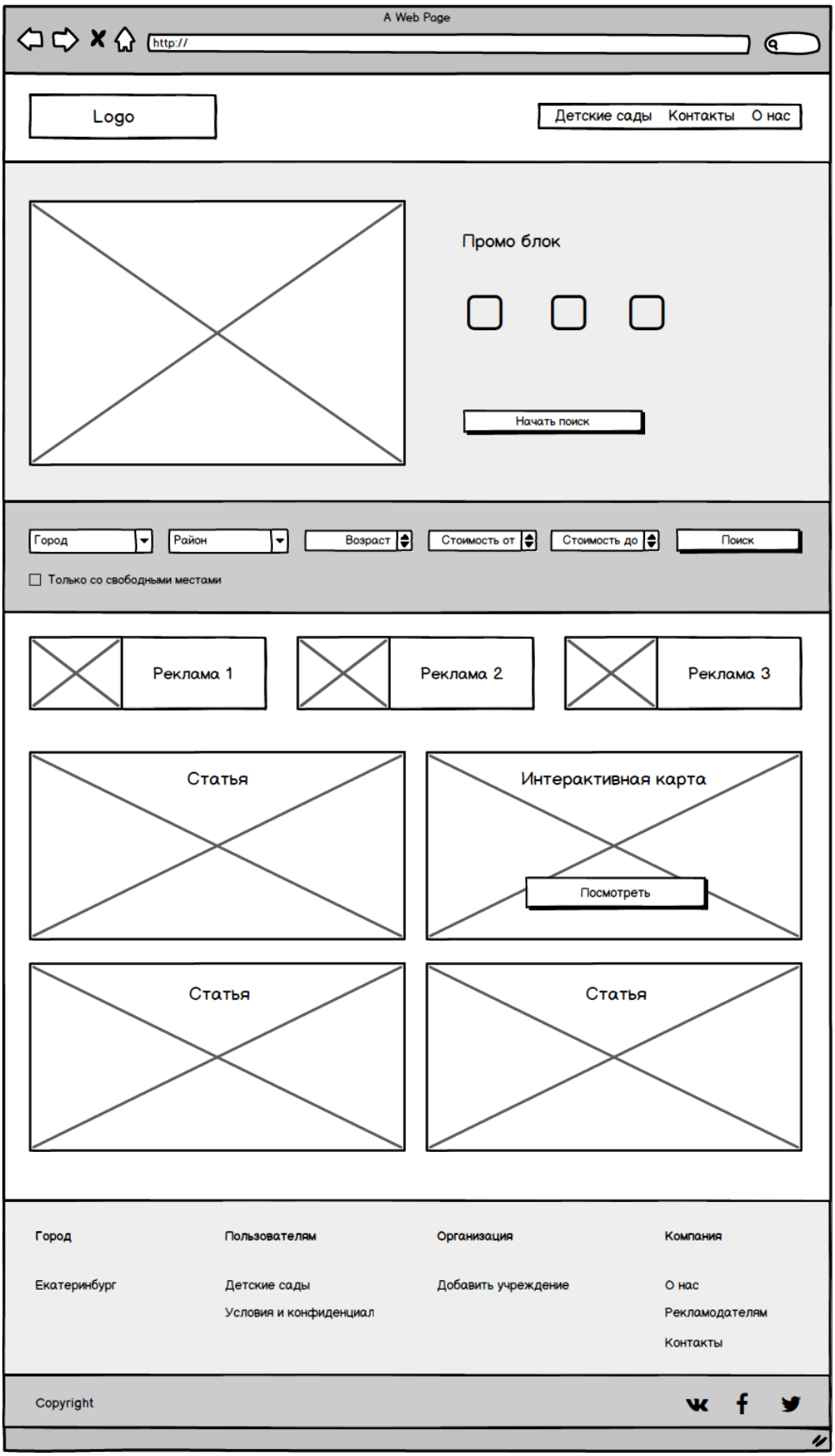

Рис. 14. Прототип главной страницы

Блок с системой фильтров является основным блоком, т.к. поиск и отбор учреждений является целевым действием пользователя. Он состоит из выпадающего списка городов – для выбора города, по которому осуществляется поиск (важен для дальнейшего развития сервиса). Выпадающий список выбора района города служит для ограничения выборки по районам города, т.к. детские учреждения выбирают исходя из близости расположения к дому. Строка ввода возраста ребенка позволяет более точно отобрать учреждения, которые принимают детей возраста, введённого в данное поле. Отбор по стоимости используется, т.к. стоимость является одним из ключевых параметров, по которым пользователь принимает решение в пользу конкретного учреждения. Флаг «Только со свободными местами» - в выборку попадают учреждения, где есть свободные места.

Рекламный блок – один из элементов, который позволит монетизировать сервис.

Блок статей – содержит ссылки на статьи с полезной информацией, которая будет интересна пользователям. Блок «Карта учреждений» ведет на страницу интерактивной карты, где пользователь может выбрать подходящее учреждение исходя из местоположения.

Подвал сайта выполнен стандартно для такого типа сайта и содержит копирайт, ссылки на группы в социальных сетях и меню, для того что бы пользователь имел возможность перейти на интересующий раздел сайта после прокрутки главной страницы до конца.

На рис. 15 представлен прототип каталога.

Эта страница содержит систему фильтров, как в блоке на главной странице, сортировку результатов поиска учреждений по стоимости и непосредственно вывод результатов.

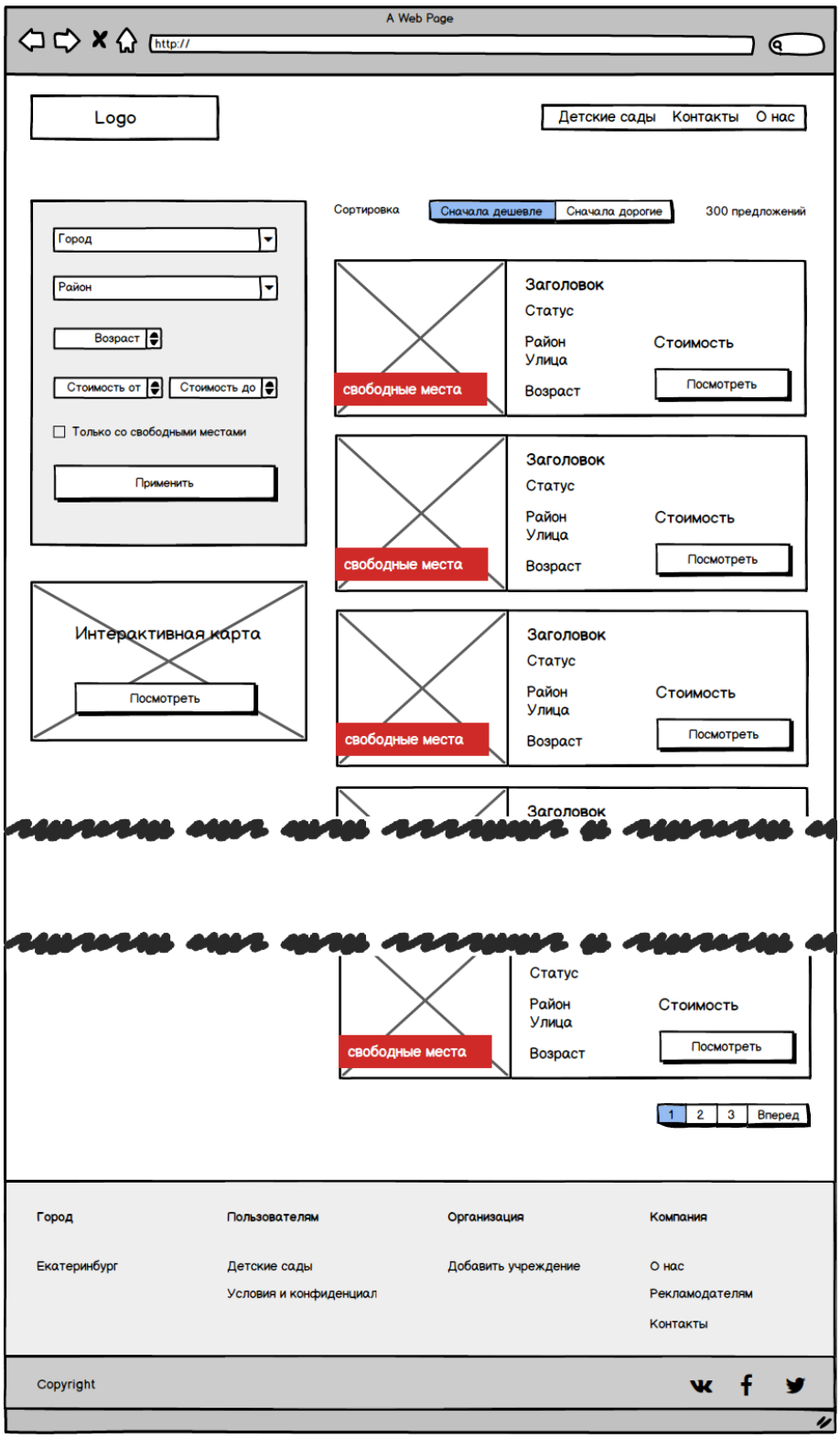

Рис. 15 Прототип страницы каталога

Блок вывода учреждения предоставляет следующую информацию:

- название учреждения;
- статус учреждения;
- район и адрес;

- возраст детей, которых принимает учреждение;
- наличие или отсутствие свободных мест;
- стоимость в месяц.

На странице выводится 8 учреждений и средства навигации для постраничного просмотра результатов отбора. Такое количество оптимально с точки зрения длины страницы и восприятия информации пользователями.

На рис. 16 представлен прототип страницы учреждения.

Страница содержит информацию и описание детского сада, стоимость и интерактивную карту. При нажатии на кнопке «Записаться в группу» появляется модальное окно с формой, рис. 17, в которой необходимо ввести данные. При отправке формы приходит сообщение администратору сервиса с данными пользователя, после проверки которых, администратор уведомляет менеджера учреждения о том, что поступила заявка на запись в данное учреждение. Так же данные фиксируются в панели управления сервисом. Пользователю приходит сообщение на почтовый ящик, который он указал в форме записи, с благодарностью за использование сервиса и информацией о дальнейших действиях.

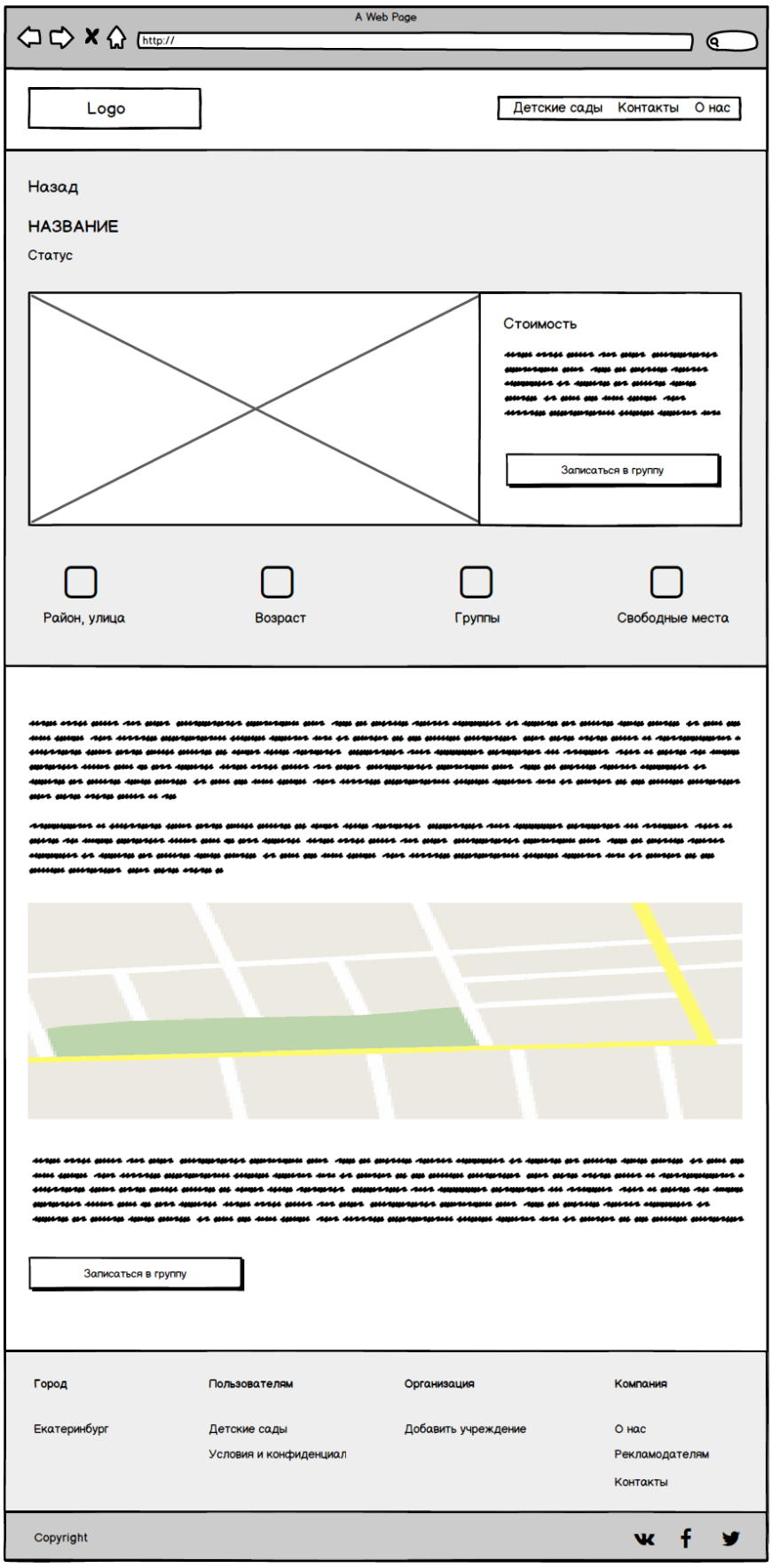

Рис. 16 Прототип страницы учреждения

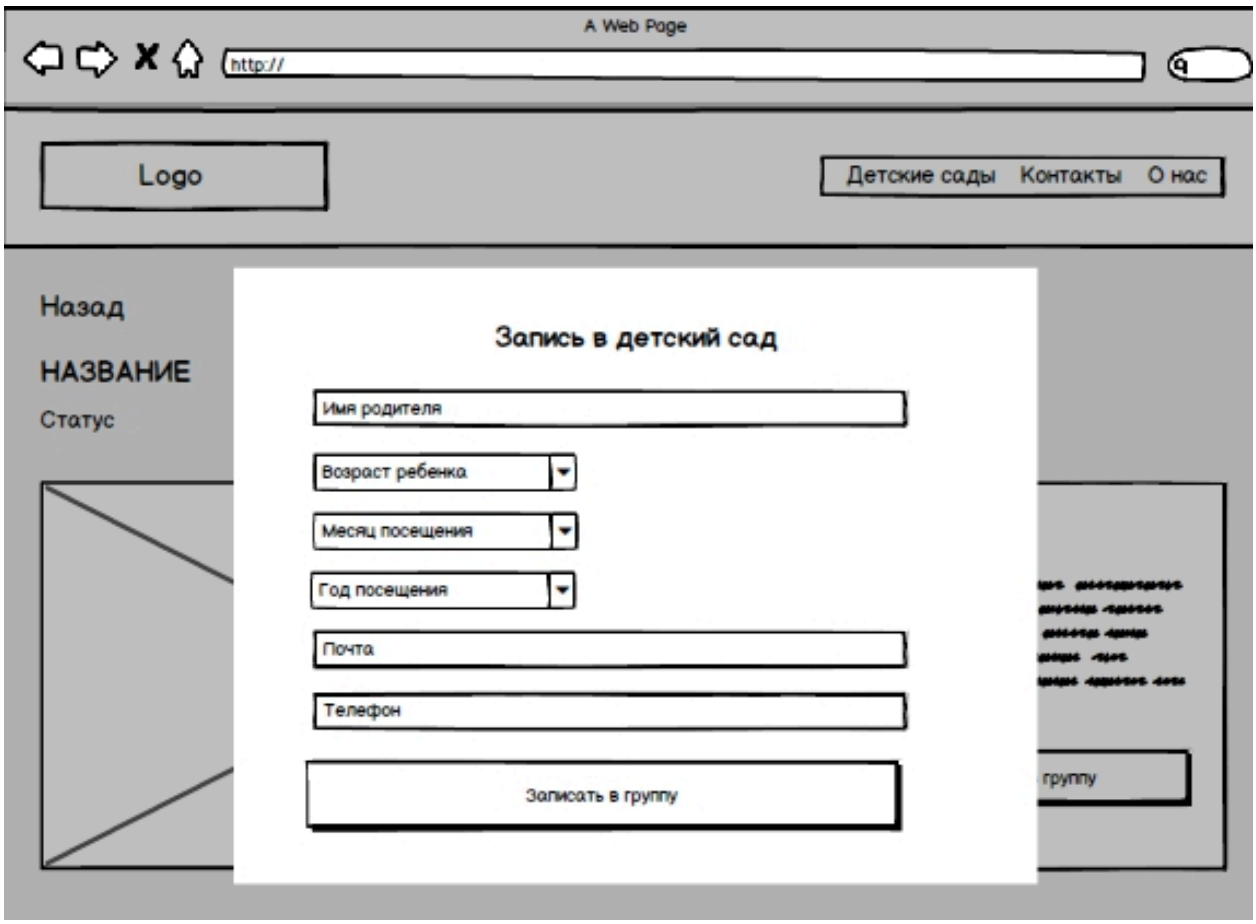

Рис. 17 Модальное окно с формой записи в учреждения

Прототип были протестирован с привлечением 15 человек, подходящих под описание целевых пользователей. Большинство пользователей дали положительную оценку прототипу, отметив простоту использования и ясность представления информации. Некоторые пользователи хотели видеть отзывы об учреждении на соответствующей странице, этот функционал будет внедрен при дальнейшем развитии сервиса.

Имея прототипы сайта можно переходить к следующему этапу – созданию макетов дизайна сайта.

Разработка дизайн макетов

Для создания дизайн макетов будущего сайта используется программное обеспечение Adobe Photoshop входящее в пакет программ Adobe Creative Cloud. Adobe Creative Cloud представляет из себя набор программ, облачного сервиса и ресурсов [29].

В состав Creative Cloud входят следующие программы:

- Photoshop CC - редактирование и компоновка изображений;
- Illustrator CC - векторная графика и иллюстрация;
- InDesign CC - дизайн страниц, создание макетов и публикация;
- Dreamweaver CC - создание веб-сайтов, приложений и написание кода;
- Aftereffects CC - кинематографические визуальные эффекты и видеографика;
- Adobe Premiere Pro CC - создание и монтаж видеоматериалов;
- Adobe Muse CC - Дизайн веб-сайтов без написания кода.

Кроме этого, в Creative Cloud входят также Acrobat XI Pro, Adobe Audition CC, Bridge CC, Encore, Fireworks, Flash Builder Premium, Flash Professional CC, InCopy CC, Lightroom, Media Encoder CC, Prelude CC, SpeedGrade CC.

При создании дизайн макетов страниц web-сайта использовался Adobe Photoshop CC из пакета Adobe Creative Cloud.

Adobe Photoshop – это самый мощный на сегодняшний день графический редактор. Возможности этой программы охватывают весь спектр различных операций, связанный с графикой, а именно: обработка фотографий, создание собственных рисунков, создание постеров, коллажей, обложек для разной продукции, создание открыток и многое другое [30].

Photoshop содержит в себе сотни инструментов, тысячи функций и миллион эффектов. Интерфейс в программе простой и понятный в обращении, всегда есть возможность установить множество новых эффектов, кистей, градиентов и узоров. Интерфейс Adobe Photoshop CC представлен на рис. 18.

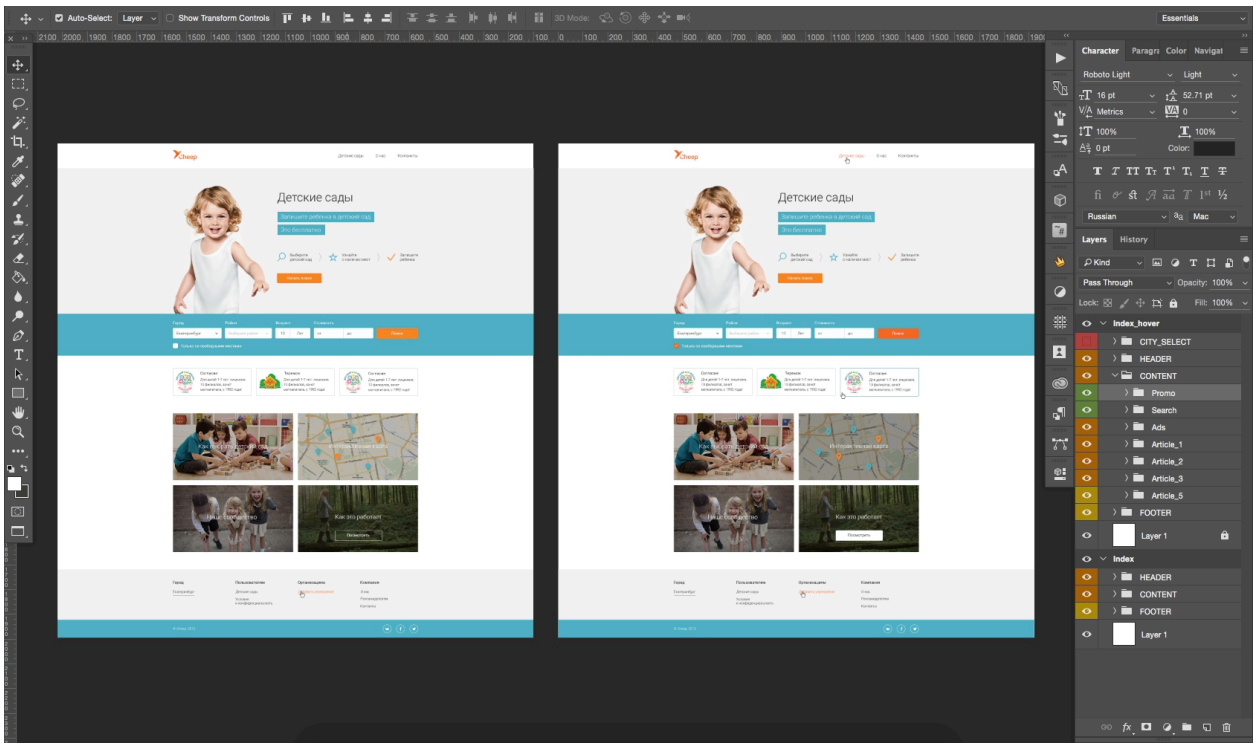

Рис. 18 Интерфейс Adobe Photoshop CC

Дизайн макеты создаваемого сервиса выполнялись строго по прототипу.

Акцент был сделан на то, чтобы сайт был прост в использовании, приятен пользователю и оставлял положительное впечатление от использования сервиса.

Результат этого этапа представлен на рис. 19-21, где изображены основные страницы сайта.

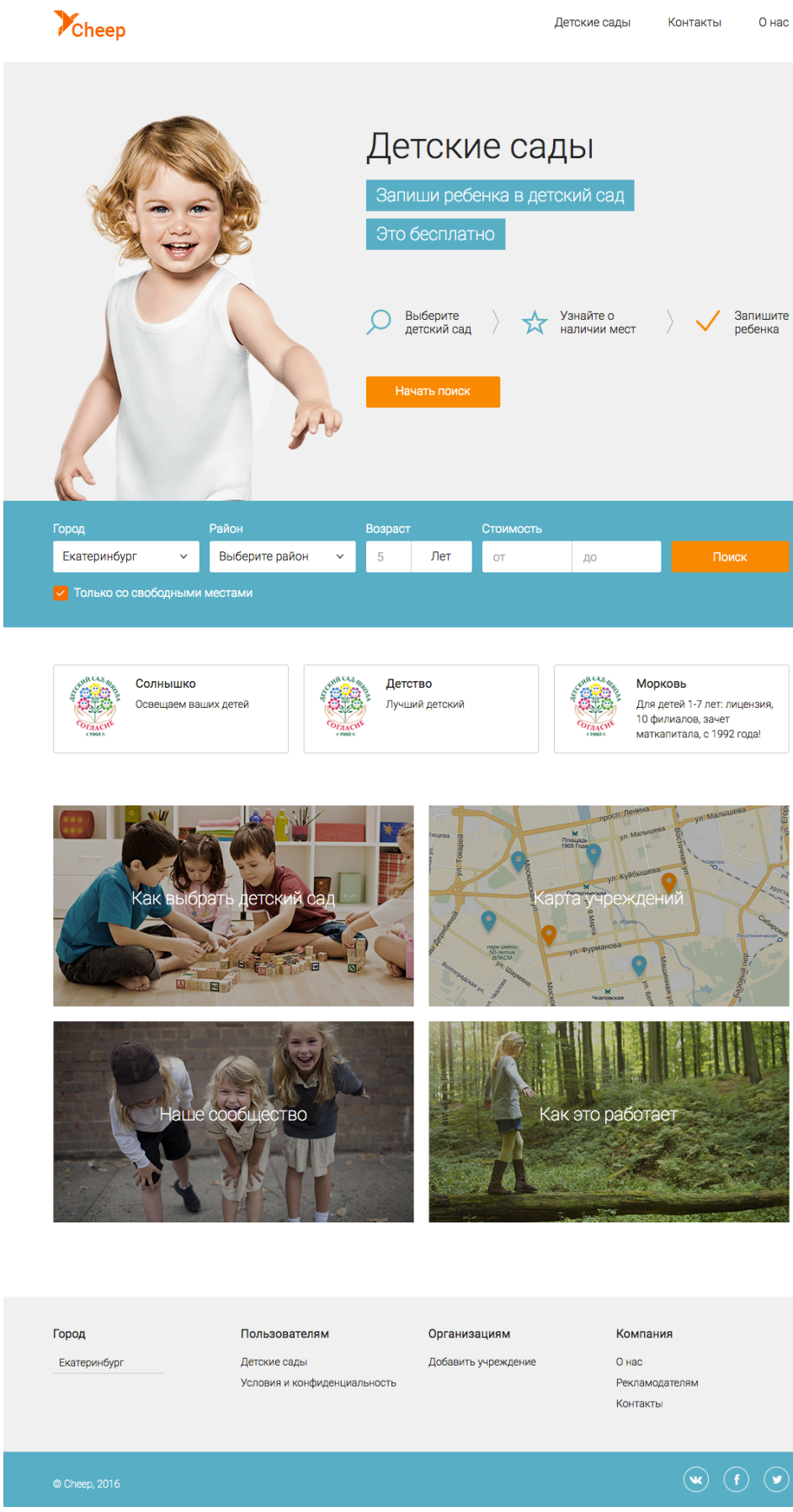

Рис. 19 Дизайн главной страницы

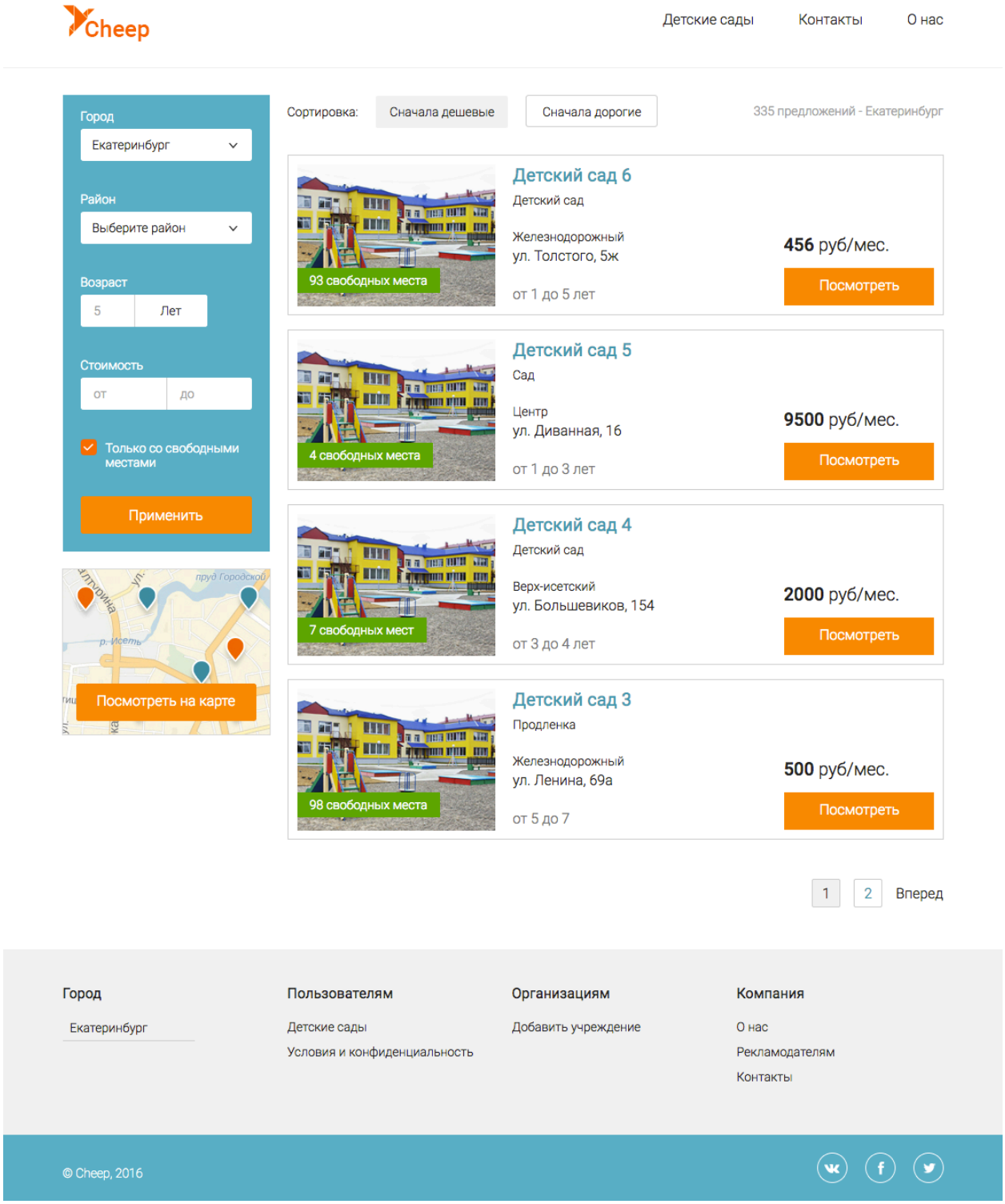

Рис. 20. Дизайн страницы каталога

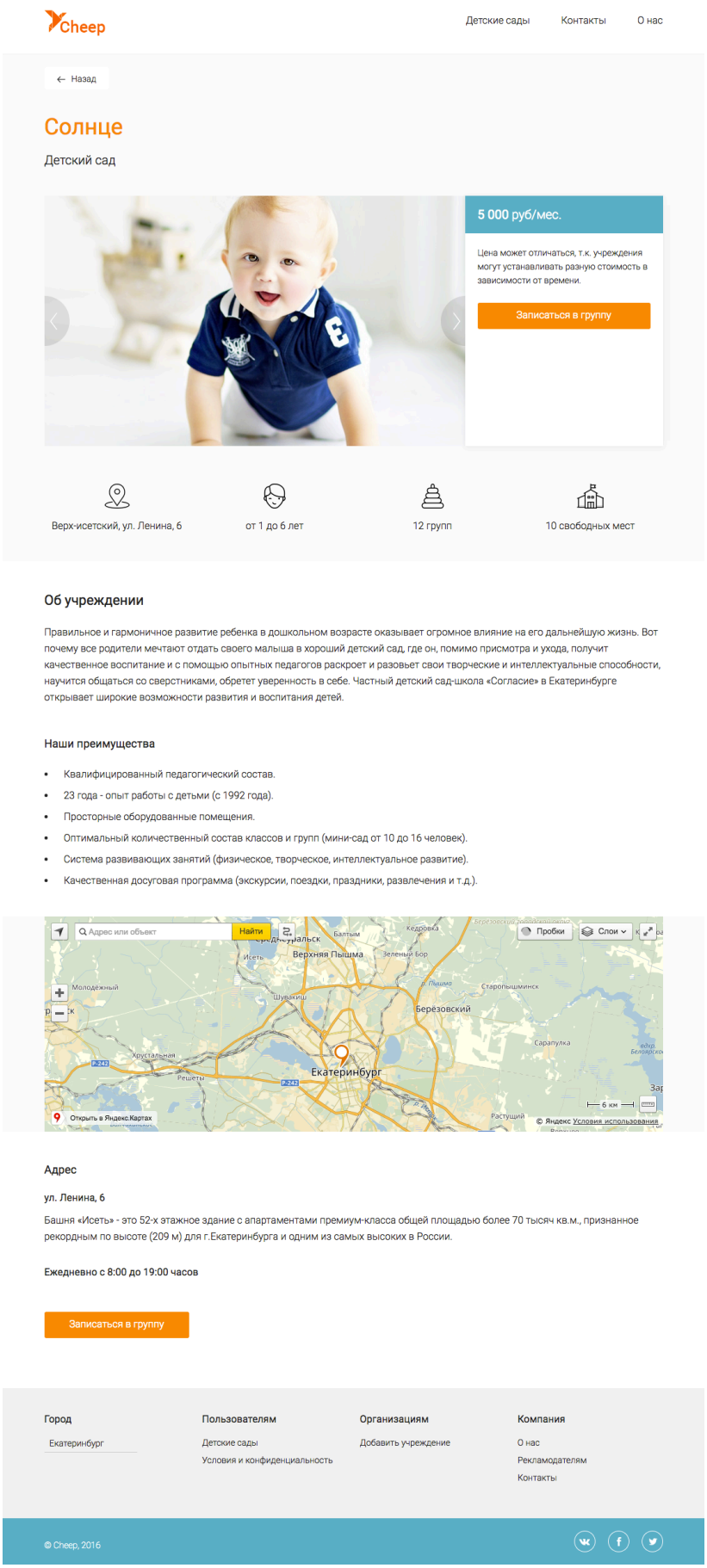

Рис. 21 Дизайн страницы учреждения

Верстка сайта

Верстка сайта представляет собой описание кодом визуальной части веб-сайта. Независимо от того, какой браузер использует пользователь, сайт должен выглядеть и работать корректно.

Существует много инструментов для верстки веб-страниц, все они делятся на три типа:

- визуальные редакторы, не требующие знаний html, css и прочих технологий для разметки страниц. В визуальном редакторе пользователь располагает различные элементы сайта, как будто на листе бумаги, а редактор пишет код самостоятельно. Однако, следует заметить, что ни один визуальный редактор не совершенен и все они так или иначе ограничены в своих возможностях, поэтому для профессиональной работы требуется умение писать код, именно поэтому нужны текстовые редакторы;
- в текстовых редакторах код пишет сам пользователь. В них, как правило, бывают разные функции облегчающие написание кода, такие, как подсветка кода (так легче видеть, где в коде вставлены стили, или скрипты, а где просто текст), различные горячие кнопки и клавиши, которые вставляют уже готовые конструкции (части кода, спецсимволы) в код, и т.д.;
- использование готовых фреймворков наподобие Bootstrap. Bootstrap - свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Для верстки сайта не использовались готовые фреймворки, написание кода осуществлялось в текстовом редакторе Brackets. Интерфейс этой программы представлен на рис. 22. Brackets – это текстовый редактор от компании Adobe. Brackets бесплатный, распространяется на все платформы и имеет продуманный и лаконичный внешний вид. В него включены разные темы оформления и возможность разбиения экрана на несколько частей. Эта про-

грамма имеет множество расширений, добавляющих необходимые инструменты для работы над кодом, таких как система контроля версий Git, просмотр HTML-кода в браузере в реальном времени (Live Preview), синхронизация с FTP и другие.

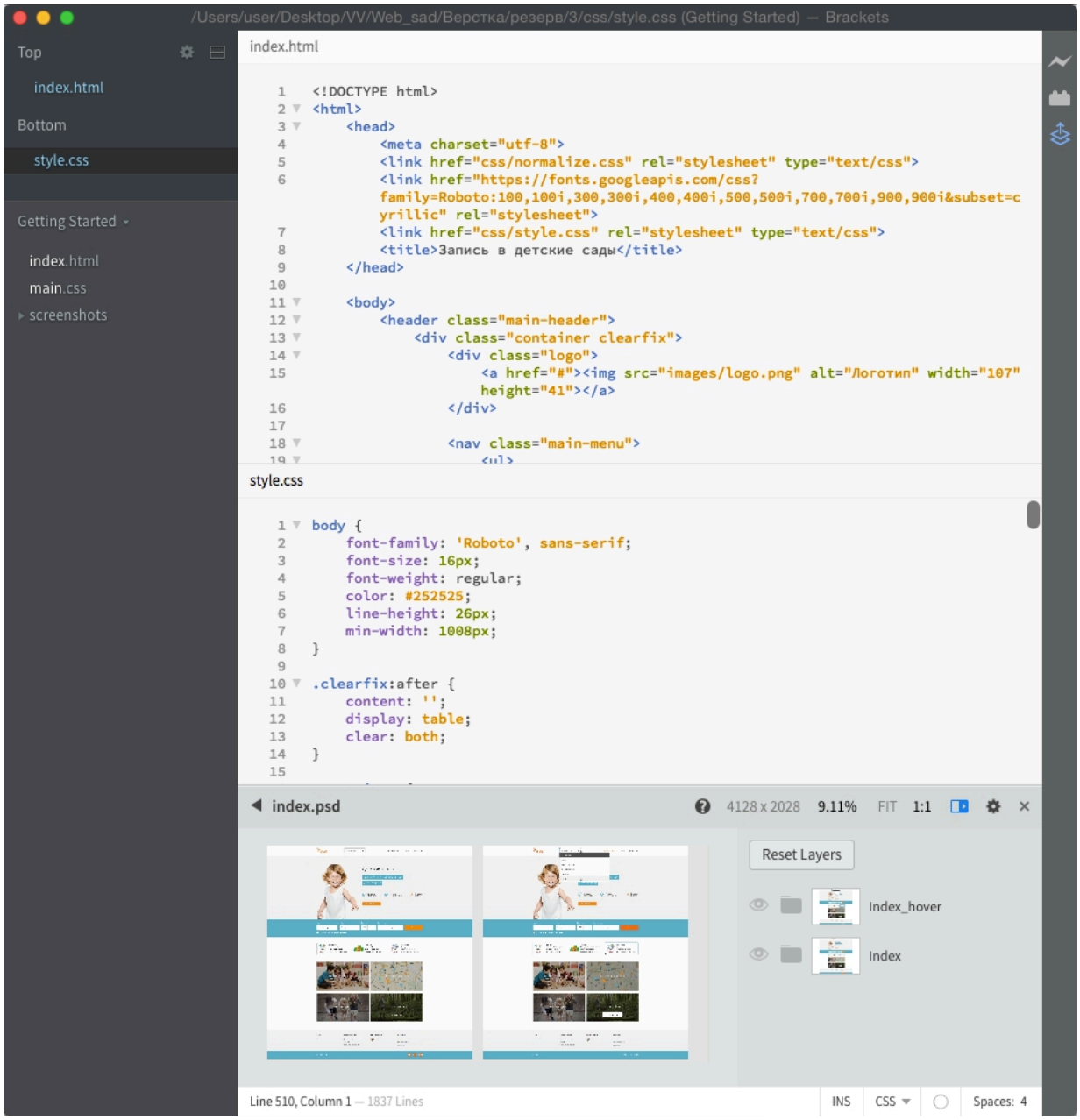

Рис. 22 Интерфейс Brackets

Выбор платформы для создания сервиса

От выбора платформы для разработки зависит как сама по себе реализуемость необходимого функционала, так и возможности дальнейшего развития веб-проекта. Для создания сервиса, как правило, выбирается одна из

платформ: CMS, фреймворк или SaaS-решение.

Система управления сайтом (Content Management System) — это программный продукт, который служит для разработки различных разновидностей веб-сервисов. Почти все CMS модульные, а модули многих из них собраны в комплекты (или редакции), предназначенные для тех или иных видов сайтов.

CMS - это специальная программа, которая устанавливается на хостинг-площадке и которая выполняет две основные функции.

Главная функция CMS - показывать страницы сайта пользователям, динамически формируя их содержимое из заранее определенных шаблонов с дизайном и контентом, то есть текстов, картинок, таблиц и других материалов, которые хранятся в базе данных.

По своей сути, CMS является программной реализацией платформы, включающая в себя технологии для разработки веб-приложений.

Вторая функция CMS - помочь владельцу сайта без каких-то специальных навыков управлять сайтом, то есть публиковать новые страницы, новости, выкладывать видео, делать ссылки на внешние ресурсы и так далее.

Фреймворк — это программный продукт, который служит основой для сервиса, но обычно не содержит в себе готовых программных модулей для реализации конкретных процессов. Выражаясь техническим языком, фреймворк — это более низкоуровневое решение, нежели CMS. Разработчики, при создании сервиса на фреймворке, создают не только публичную часть сайта, но и проектируют базу данных, разрабатывают алгоритмы для модулей системы, а также создают административный интерфейс для управления проектом. Необходимость серьёзных затрат на программирование делает разработку более дорогой, но и результат получается более индивидуальным.

SaaS-платформа для создания сервисов — это возможность запустить достаточно простой веб-проект очень быстро и крайне дёшево (на условиях аренды). Решение подходит для простых сайтов, временных проектов и для

проверки бизнес-идей.

Наш сервис будет строиться на основе системы управления сайтом, т.к. структура сервиса является стандартной для данного типа проектов и CMS соответствуют всем требованиям, необходимым для разработки современного веб-сервиса.

Выбор системы управления

На сегодняшний день существует широкий выбор систем управления сайтами. Компания iTrack предоставила независимый рейтинг систем управления сайтом, рис. 23, составленный по информации о реальных установках на сайтах. Было опрошено 4 901 485 доменов зоны RU [31].

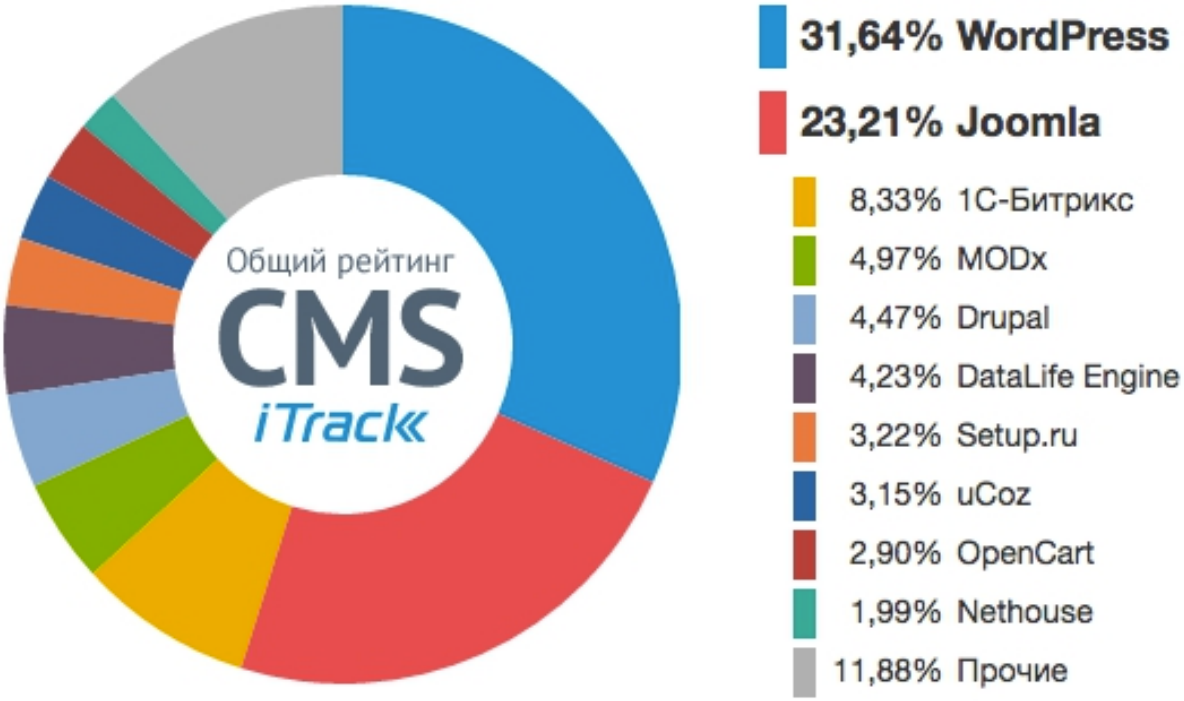

Рис. 23 Рейтинг CMS

В нашей работе будем использовать одну из популярных CMS, т.к. они обладают рядом преимуществ:

- высокий профессиональный уровень разработчиков CMS, который находит своё отражение в логичной и понятной архитектуре системы;
- наличие необходимой документации;
- наличие большого сообщества разработчиков, что в случае проблем с системой позволяет оперативно найти ответ на любой вопрос;

- наличие большого количества дополнительных модулей, расширяющих функциональность сервиса.

Для разработки веб-сервиса была выбрана система **MODX Revolution**.

Проведя анализ данной системы были выделены ее основные преимущества и недостатки.

MODX — это бесплатная профессиональная система управления содержимым (CMS) и фреймворк для веб-приложений, предназначенная для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым) сайтов [32].

MODX распространяется бесплатно по лицензии GPL с открытым исходным программным кодом (Open Source). Это означает, что систему MODX может использовать каждый: как для личного использования, так и для коммерческого распространения сайтов, построенных на данной системе управления.

MODX написана на программном языке PHP и использует для хранения данных СУБД MySQL или MS SQL. Система управления MODX может быть установлена на большинстве веб-серверов.

Преимущества MODX Revolution:

- универсальность - подходит для создания интернет-проектов различного назначения;
- высокая скорость работы достигается благодаря хорошо продуманной архитектуре ядра, системам кеширования и шаблонизации;
- гибкость - система шаблонов позволяет полностью контролировать исходный код;
- SEO-ориентированность - идеально подходит для продвижения сайта в интернете и изначально оптимизирована под поисковые системы;
- удобная панель управления позволяет сделать простой доступ к любому разделу, создаёт древовидную структуру документов с неограниченным уровнем вложенности, позволяет работать с документами на сервере;

- масштабируемость - система представляет собой идеальное сочетание системы управления и фреймворка, имеется внушительный список готовых, грамотно написанных и гибко настраиваемых дополнений, которые легко устанавливать и обновлять;
- доступность – бесплатная система с открытым исходным кодом;
- MODX является современной системой, соответствующей идеологии web 2.0, в полной мере использующей AJAX.

Недостатки MODX Revolution:

- необходимо иметь значительный опыт разработки веб-сервисов;
- недостаточное количество документации на русском языке.

Разработка сервиса на платформе MODX

Для создания сервиса на MODX Revolution необходимо убедиться, что конфигурация сервера соответствует минимальным системным требованиям [33]. А именно:

- операционная система - Linux x86, x86-64, Windows XP или Mac OS X;
- веб-сервер - Apache 1.3.x / Apache 2.2.x, IIS выше 6.0, Zeus, Cherokee, lighthttpd или nginx;
- базы данных — MySQL 4.1.20 или выше (исключая версию 5.0.51), либо Microsoft SQL Server 2008. Кодировка таблиц по умолчанию — UTF-8. Плюс должны быть разрешены команды SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, INDEX, DROP. Дополнительно должен поддерживаться механизм хранения под названием MyISAM;
- PHP и его модули — PHP версии 5.1.1 или выше (исключая 5.1.6/5.2.0), запуск с FastCGI. Должны быть установлены FastCGI, JSON, cURL, Imagemagick, GD lib, PDO с драйвером баз данных, SimpleXML. Также настройки php.ini должны быть следующими: safe_mode off, register_globals off, magic_quotes_gpc off. Плюс memory_limit не меньше 24 МБ.

Самым простым способом установки MODX на локальную машину яв-

ляется использование установщика Bitnami, доступного по адресу https://bitnami.com/stack/modx. Инсталлятор автоматически развернет всю необходимую инфраструктуру для работы с системой MODX – MySQL Database (система управления базами данных) и Apache Web Server.

Первым делом необходимо создать базу данных и пользователя. Сделать это можно через панель управления базой – phpMyAdmin, рис 24.

При создании базы ее кодировку и сопоставление необходимо установить utf8 и utf_general_ci соответственно.

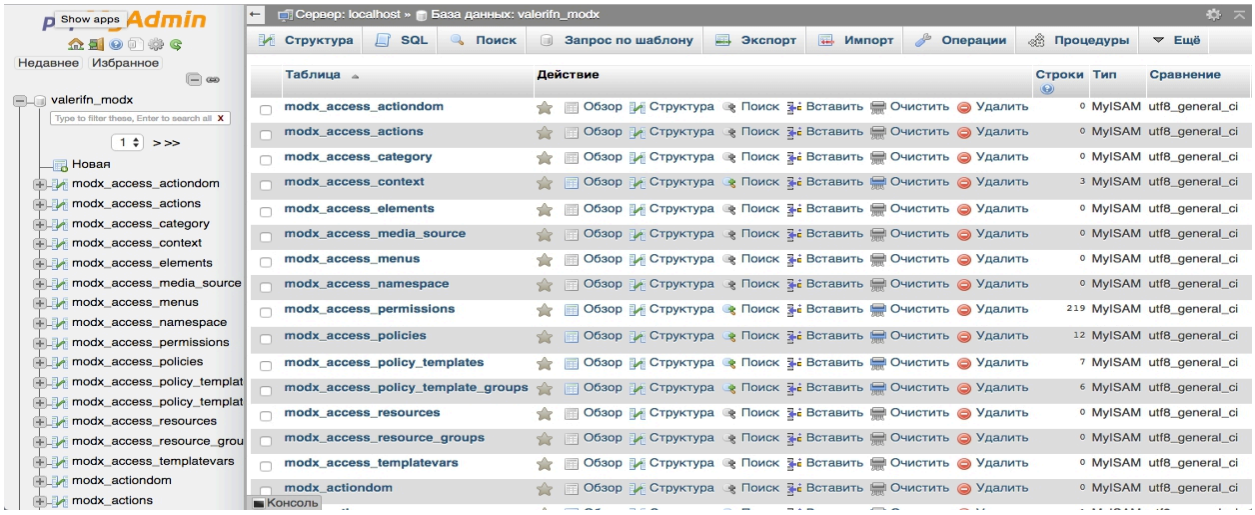

Рис. 24 Панель управления БД и созданная база

Далее необходимо соединить систему MODX с созданной базой данных, введя в панели управления MODX имя базы, пользователя, пароль, хост и префикс. Затем создаем администратора и после этого можем осуществить вход в панель управления MODX, рис. 25.

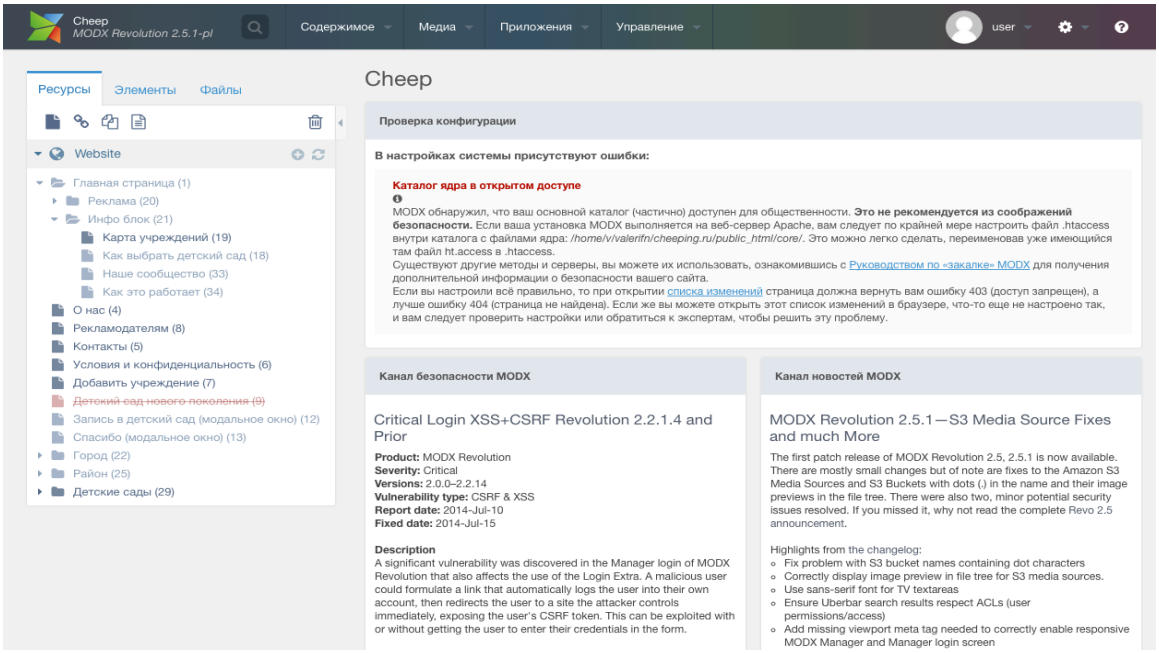

Рис. 25 Панель управления MODX

Базовыми элементами в системе управления являются шаблоны, ресурсы, переменные шаблона, чанки, сниппеты и плагины, поэтому кратко опишем каждый элемент.

MODX шаблоны являются основой для веб-страницы. Как правило, не содержит ничего, кроме HTML-кода и MODX-тегов. Некоторые шаблоны содержат также код JavaScript. При запросе документа, MODX загружает документ и шаблон, и MODX находит все специальные заполнители в шаблоне и заменяет их с соответствующими значениями из документа перед отправкой готовую страницу в браузер пользователя [34]. В то время когда документ запрашивается, MODX грузит документ в шаблон и все особые поля в шаблоне сменяет на соответствующие значения из документа, перед отправкой страницы браузеру пользователя. Для организации шаблона необходимо перейти на вкладку элементы, и нажать «Новый шаблон». Заполнить необходимые поля и ввести HTML код в поле "Код шаблона".

Ресурсом называется представление страницы в MODX. Существуют разные виды ресурсов: документ, веб-ссылка, символическая ссылка и статический ресурс. Ресурсом по умолчанию называется документ - обычное представление страницы вашего сайта. Любой ресурс располагает уникаль-

ным номером ID. По этого номеру MODX определяет, какой именно ресурс нужно загружать. Если необходимо поставить ссылку на ресурс нужно использовать для этого ID. При помощи этого MODX создает ссылку, которая не зависит от типа ресурса, имени, псевдонима и т.д., таким образом, при смене параметра ресурса ссылка на ресурс все равно остается действующей. Ресурсы представлены во вкладке ресурсы в панели управления в дереве левого навигатора. Чтобы изменить содержание ресурса нужно воспользоваться текстовым полем внизу страницы.

Дополнительные поля используются для расширения атрибутов ресурса, которые доступны по умолчанию. Обыкновенные ресурсы MODX располагают некоторой численностью установленных по умолчанию полей: заголовок (pagetitle), содержание (content), описание (description) и т.д. В случае необходимости существует возможность добавления собственных полей к странице, например, второе поле содержания, либо выпадающий список, либо другие иные пользовательские данные. Это возможно благодаря дополнительным полям. MODX дает возможность располагать фактически неограниченной численностью дополнительных полей. Пример вызова дополнительного поля - [[*name]].

Чанки — это части HTML кода или текста. Важно отметить что чанк — это чистый HTML-код. Чанк не может содержать PHP-код, он просто не будет выполнятся. PHP-код необходимо вставлять в сниппет. Этот сниппет может быть вызван в чанке. Чанки могут использоваться для различных целей. Некоторые примеры использования чанков включают организацию шаблона в управляемые части, содержащие HTML или текст, который будет использован снова и снова на сайте, а также для хранения мини-шаблонов для меню и др. Пример вызова чанка - [[$chunkName]].

Сниппеты – это части PHP-кода, которые позволяют добавлять функциональность в MODX сайт. Они обеспечивают пользовательский динамичный контент, такой как меню, блоги или новые списки и другие функцио-

нальные блоки и все, что сайт должен генерировать по запросу [35]. Пример вызова сниппета - [[snippetName]].

Плагин в MODX - это PHP-код, который выполняется при наступлении определенных событий. Этот код невозможно вызвать в шаблонах, при создании плагина указывается одно или несколько из предустановленных событий, к которым привязан данный код. Таким образом, плагины расширяют функционал самого ядра MODX, позволяя не вносить изменения в код системных файлов MODX.

Разработка страницы каталога

Рассмотрим, каким образом разрабатываются страницы сервиса на примере ключевой страницы каталога учреждений.

Разработка любой страницы начинается с подготовки шаблона. Шаблон сайта содержит в себе файлы – css-файлы, скрипты, изображения и т.д. Все это нужно хранить на сервере, чтобы шаблон имел к этим файлам доступ. Таким образом, первым шагом является перенос папок с файлами шаблона на сервер, рис.26.

После загрузки папок с файлами можно начать создание MODX шаблона. В панели управления на вкладке «Элементы» создаем новый шаблон «Каталог». Html-код сверстанной страницы каталога целиком копируем в новый шаблон. Необходимо изменить пути ссылок к файлам, подключенным в html-коде, т.к. после загрузки файлов на сервер пути доступа к файлам поменялись.

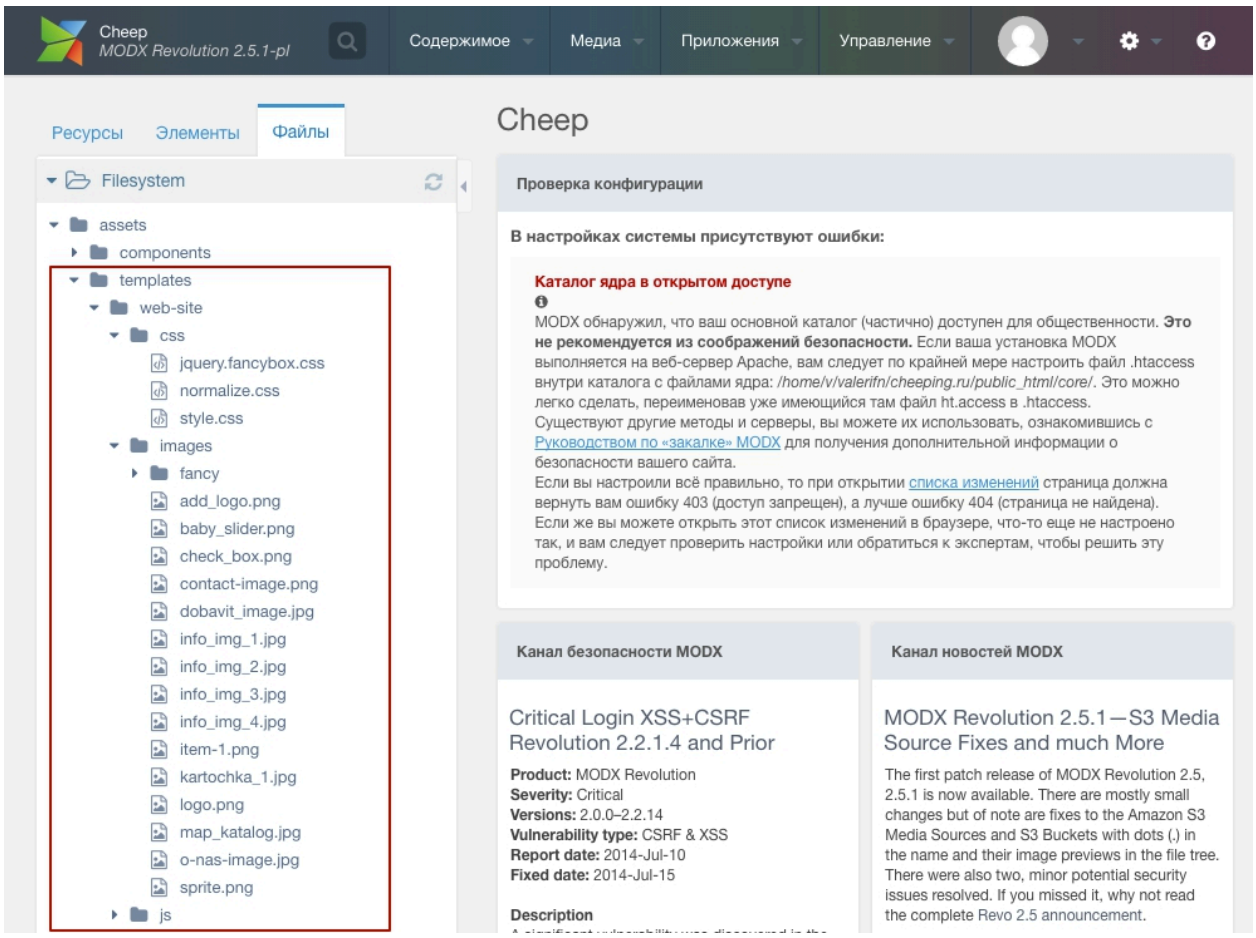

Рис. 26 Структура файлов

Изменим тег заголовка сайта так, чтобы он выводил имя сервиса, а не имя шаблона. Для этого текст в теге *<title>Запись в детские сады</title>* меняем на системный тег *<title>[[++site_name]]</title>*, который передаст текущий заголовок сайта с системных настроек. В элемент *<head>* необходимо добавить тег *<base>*, для определения адреса по умолчанию для всех ссылок на странице - *<base href="[[++site_url]]">*, в этом случае так же используем системный тег.

Проанализировав страницы сервиса, можем отметить, что верхняя часть («шапка»), содержащая логотип и меню, и нижняя часть («футер») являются одинаковыми для всех страниц. Поэтому создадим для этих элементов чанки, которые будем подключать во всех шаблонах страниц.

Каждому чанку назначаем имя, а в код чанка вставляем html-разметку блока.

Например чанк main-header-page, содержащий разметку «шапки» будет

выглядеть следующим образом:

```
<header class="main-header">
  <div class="container clearfix">
    <div class="logo">
      <a href="[[~1]]"><img src="assets/templates/web-site/images/logo.png" alt="Логотип" width="107" height="41">
      </a>
    </div>
      <nav class="main-menu">
          <ul>
            <li><a href="#" class="active">Детские сады</a></li>
            <li><a href="#">О нас</a></li>
            <li><a href="#">Контакты</a></li>
          </ul>
        </nav>
  </div>
</header>
```

Далее вызовем этот чанк в шаблоне страницы каталога, вместо html-разметки «шапки» вставляем тег вызова чанка [[$main-header-page]]. Парсер MODX считывает вызов чанка, берёт его содержимое и размещает в области, где сделан вызов.

Таким образом, делим страницы сервиса на смысловые части, которые можно разнести по чанкам и подключать в шаблонах по необходимости.

В настоящее время наше меню является статическим, и представляет из себя просто html-разметку. Для создания динамического меню используется сниппет Wayfinder. Wayfinder – это сниппет, который выводит неотсортированный список ссылок на ресурсы в дереве сайта, тип вывода списка зависит от вызова сниппета и параметров данного вызова. Когда помещается вызов Wayfinder в шаблоне, он начинает искать ресурсы, которые отвечают задан-

ным в нём параметрам и возвращает список ссылок к этим ресурсам в формате неотсортированного списка либо в определенном вами формате.

В чанк main-header-page между тегами <nav> поместим код вызова снипета Wayfinder:

```
[[!Wayfinder?
    &startId=`0`
    &includeDocs=`29,4,5`
    &sortOrder=`DESC`
    &outerTpl=`TopMenuOuter`
    &rowTpl=`TopMenuRow`
    &hereClass=`active`
     &firstClass=``
     &lastClass=``]]
```

Параметр *&startId=`0`* - начальный ID означает, что необходимо начинать с корня дерева (**ID** 0 - корень сайта) и показывать все ресурсы, которые опубликованы и у которых не стоит галочка «Спрятать от меню»**.**

Более подробную информацию о данном сниппете можно получить из официальной документации. Таким образом мы настроили вывод динамического меню, теперь при добавлении или удалении ресурса, изменения будут отображаться и в меню. Никаких дополнительных действий со стороны администратора больше не потребуется.

Теперь обратим внимание на блоки учреждений.

Чтобы вывести данные блоки необходимо произвести отбор по каталогу всех учреждений, и вывести только те, которые удовлетворяют условиям отбора.

Таким образом, необходимо сформировать каталог с учреждениями. Для начала необходимо создать шаблон страницы учреждения. Затем создать дополнительные поля с изменяемой информацией для этого шаблона. И наполнить каталог учреждениями, рис. 27.

Ресурс «Детские сады (ID-29)» является страницей вывода каталога, его шаблон Каталог.

Ресурсы «Согласие (ID-30)» и остальные ресурсы, по иерархии расположенные внутри ресурса «Детские сады (ID-29)», являются страницами учреждения. Их шаблоном является шаблон Учреждение с подключенными дополнительными полями, а в дополнительных полях самих ресурсов указываются данные конкретного учреждения.

Таким образом формируем каталог учреждений.

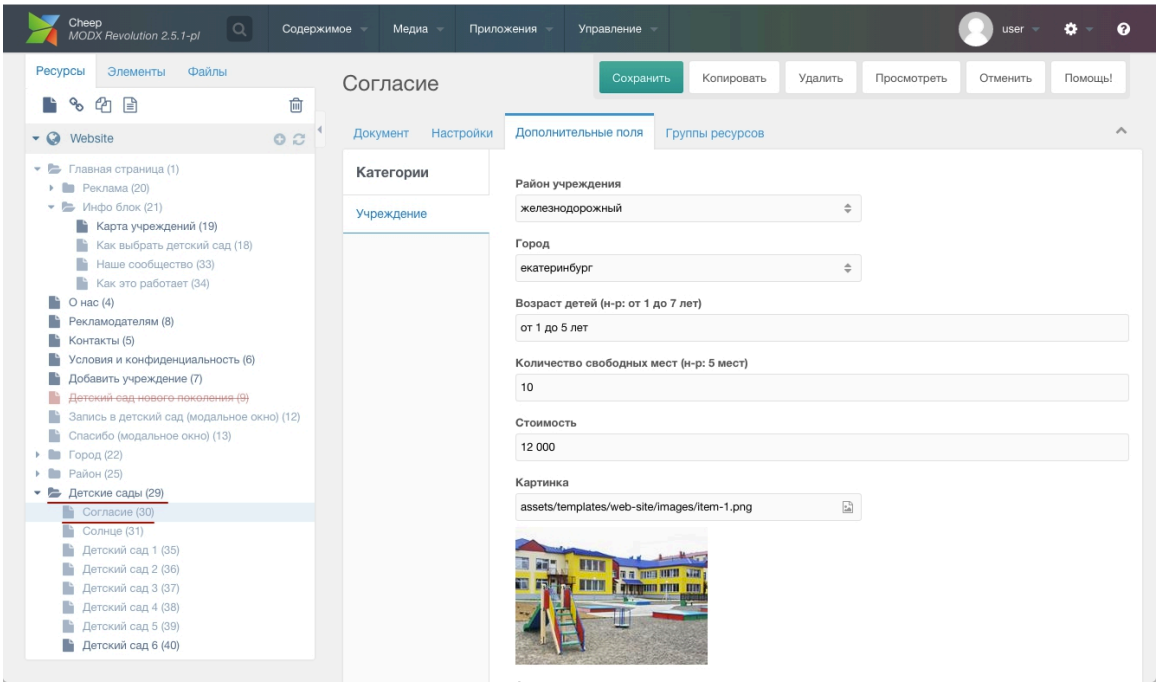

Рис. 27 Каталог детских садов

Каталог с учреждениями создан, теперь для того, что бы вывести блоки из каталога учреждений на страницу Каталог. Используем сниппет getResources. getResources используется для случаев, где необходимо объединить и вывести информацию от различных ресурсов в одном месте и в необходимом формате. Вызов сниппета осуществляется в шаблоне Каталог:

```
[[!pdoPage@katalog-pdo?
  &element=`getResources`
  &showHidden=`1`
  &tpl=`tplList`
  &limit=`8`
```

&includeContent=`1`

&includeTVs=`1`

&processTVs=`1`

&pageLimit=`5`

&pageNavVar=`page.nav`

]]

* *

[[!+page.nav]]

* *

Основными параметрами данного сниппета являются *&tpl=`tplList`* и *&limit=`8`*. *&tpl=`tplList`* - определяет чанк для вывод содержимого ресурсов, служащий для того, что бы страница выводилась в нужном виде.

&limit=`8` - указывает максимальное количество блоков, которые выводятся на странице. Более подробную информацию о данном снипете можно получить из официальной документации. Для разбиения контента на страницы и добавления навигации по ним используется сниппет *pdoPage.*

Таким образом, осуществляется вывод блоков учреждений на странице каталога.

Теперь добавим сортировку по стоимости и фильтрацию.

В сниппете getResources за сортировку отвечают параметры:

- **&sortby** – сортировка по какому-либо полю ресурса (кроме tv-параметров);
- **&sortbyTV** – сортировка по tv-параметру;
- **&sortdirTV** – направление сортировки по tv-параметру (desc|asc);
- **&sortbyTVType** – тип сортировки (string|integer).

В параметре **&sortbyTVType** прописываем **`integer`**, так как стоимость это число. В **&sortdirTV** нам нужно записывать ASC или DESC в зависимости от get-параметра **sortdir**. Напишем сниппет *[[!sortbyTV]]*.

```
<?php
$sort_direction = (isset($_GET['sortdir']) && $_GET['sortdir'] == 'asc') ? 'ASC' : 'DESC';
return $sort_direction;
```

Код вызова сниппета getResources выглядит следующим образом:

```
[[!pdoPage@katalog-pdo?
  &element=`getResources`
  &showHidden=`1`
  &tpl=`tplList`
  &limit=`8`
  &includeContent=`1`
  &includeTVs=`1`
  &processTVs=`1`
  &pageLimit=`5`
  &pageNavVar=`page.nav`
  &sortby=`{[[!sort]]"publishedon":"DESC"}`
  &sortbyTV=`[[!sortbyTV]]`
  &sortdirTV=`[[!sortdirTV]]` &sortbyTVType=`integer`
]]

<div class="pagination">
[[!+page.nav]]
</div>
```

За фильтрацию по tv-параметрам в getResources отвечает параметр &tvFilters. Пишем сниппет *[[!filter]]* для формирования &tvFilters из get-параметров:

```
<?php
$out = array();
$city = isset($_GET['city']) ? strip_tags($_GET['city']) : '';
```

```
$rayon = isset($_GET['rayon']) ? strip_tags($_GET['rayon']) : '';
$vozrast = isset($_GET['vozrast']) ? strip_tags($_GET['vozrast']) : '';
$price = isset($_GET['price']) ? strip_tags($_GET['price']) : '';
if ($city != '') $out[] = 'city==' . $city;
if ($rayon != '') $out[] = 'rayon==%' . $rayon . '%';
if ($vozrast != '') {
foreach ($categories as $categ) {
$out[] = 'category==%*' . $categ . '*%';
}
}
return implode(',', $out);
```

Окончательный код вызова сниппета getResources выглядит следующим образом:

```
[[!pdoPage@katalog-pdo?
  &element=`getResources`
  &showHidden=`1`
  &tpl=`tplList`
  &limit=`8`
  &includeContent=`1`
  &includeTVs=`1`
  &processTVs=`1`
  &pageLimit=`5`
  &pageNavVar=`page.nav`
  &sortby=`{[[!sort]]"publishedon":"DESC"}`
  &sortbyTV=`[[!sortbyTV]]`
  &sortdirTV=`[[!sortdirTV]]` &sortbyTVType=`integer`
  &tvFilters=`[[!filter]]`
]]
<div class="pagination">
```

[[!+page.nav]]

* *

Таким образом, на странице каталога происходит вывод и фильтрация контента.

Подобно странице каталога были разработаны и остальные страницы сервиса.

Сервис выполнен с учетом последних тенденций веб-дизайна и технологий, при разработке сервиса были пройдены все этапы от исследования конкурентов и целевой аудитории до запуска проекта.

ЗАКЛЮЧЕНИЕ

Целью данной работы являлась разработка корпоративной информационной системы для записи детей в детские сады. При выполнении работы были рассмотрены технологии, которые используются для организации современных веб-сервисов, и методология проектирования информационных систем. Используя теоретическую базу и лучшие практики были решены все поставленные задачи, а именно:

1) проанализирована предметная область,
2) спроектировано веб-приложение,
3) создана дизайн-концепция, осуществлена верстка страниц,
4) разработаны функциональные инструменты и интегрированы в систему управления содержимым,
5) материалы размещены на сайте и произведена пробная эксплуатация.

В результате решения поставленных задач проект был успешно запущен.

В дальнейшем предполагается масштабирование сервиса на другие города России, расширение функционала, например, добавление отзывов об учреждениях. Но предварительно будет осуществлен этап сбора информации от пользователей сервиса, мнений и пожеланий. Исходя из результатов работы за отчетный период времени, будет корректироваться стратегия и путь дальнейшего развития сервиса.

Проект имеет социальное значение как для родителей, так и для самих детских садов, учитывая сложившуюся ситуацию с муниципальными учреждениями, в которых отсутствие мест и очереди, длящиеся по нескольку лет, стали нормой. Выступая посредником между родителями и детскими садами и предоставляя равные условия для всех участников, сервис способствует развитию рынка частных детских садов, повышая доступность и качество предоставляемых услуг для рядовых граждан.

Сервис позволяет родителям совершать поиск дошкольных учреждений по ряду параметров, такие как местоположение, стоимость и возраст ребенка. Сокращает время поиска детских садов, предоставляя исчерпывающую информацию о дошкольных учреждениях.

Важность проекта для самих учреждений заключается в появлении нового канала привлечения клиентов.

Монетизация сервиса предполагается посредством размещения контекстной рекламы, приоритетной выдачи в поиске и получения комиссионного вознаграждения за предоставление клиентов детским садам.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Adobe Creative Cloud // «Adobe»: [сайт]. [2016]. URL: http://www.adobe.com/ru/creativecloud.html (дата обращения: 05.08.2016).

2. MODx // «Викиучебник»: [сайт]. [2015]. URL: https://ru.wikibooks.org/wiki/MODx (дата обращения: 09.08.2016).

3. Антамошкин О. А. Программная инженерия. Теория и практика. Красноярск: Сиб. федер. ун-т, 2012. 247 с.

4. Березднвин В. В. Управленческий учет в инвестиционном банке // «Корпоративный менеджмент»: [сайт]. [2016]. URL: http://www.cfin.ru/press/afa/2000-2/04-2.shtml (дата обращения: 16.07.2016).

5. Болодурина И.П., Волкова Т.В. Проектирование компонентов распределенных информационных систем. Оренбург: ОГУ, 2012. 215 с.

6. В России в прошлом году родилось 1,947 млн детей - впервые в современной истории // «ТАСС»: [сайт]. [2015]. URL: http://tass.ru/obschestvo/1744769 (дата обращения: 20.07.2016).

7. Гафурова Н.В., Чурилова Е.Ю. Методика обучения информационным технологиям. Теоретические основы. Красноярск: Сиб. федер. ун-т, 2012. 171 с.

8. Грошев А.С. Информатика: учебник для вузов. М.-Берлин: Директ-Медиа, 2015. 484 с.

9. Документация по MODX Revolution // «Promo Creative»: [сайт]. [2016]. URL: http://promo-creative.com/modx/ (дата обращения: 18.08.2016).

10. Документация по использованию сниппетов в системе MODX // «MODX Docs»: [сайт]. [2016]. URL: https://docs.modx.com/evolution/1.0/developers-guide/snippets (дата обращения: 20.08.2016).

11. Дружинин Г.В., Сергеева И.В. Эксплуатационное обслуживание информационных систем. М.: ФГБОУ «Учебно-методический центр по образованию на железнодорожном транспорте», 2013. 220 с.

12. Избачков Ю. С., Петров В. Н., Васильев А. А., Телина И. С. Информационные системы. 3-е изд. СПб.: Питер, 2011. 544 с.

13. Каталог частных детских садов // «2ГИС»: [сайт]. [2016]. URL: https://2gis.ru/ekaterinburg/search/%D1%87%D0%B0%D1%81%D1%82%D0%BD%D1%8B%D0%B5%20%D0%B4%D0%B5%D1%82%D1%81%D0%BA%D0%B8%D0%B5%20%D1%81%D0%B0%D0%B4%D1%8B/tab/firms?queryState=zoom%2F11 (дата обращения: 01.08.2016).

14. Коробова Л. А., Медведкова И. Е., Абрамов Г. В. Проектирование информационных систем. Воронеж: ВГУИТ, 2012. 172 с.

15. Кузнецова Л.В. Лекции по современным веб-технологиям. М.: Национальный Открытый Университет «ИНТУИТ», 2012. 165 с.

16. Купер А., Рейман Р., Кронин Д., Носсел К. Интерфейс. Основы проектирования взаимодействия. СПб.: Питер, 2016. 720 с.

17. Макарова Т.В. Компьютерные технологии в сфере визуальных коммуникаций. Работа с растровой графикой в Adobe Photoshop. Омск: ОмГТУ, 2015. 240 с.

18. Милехина О. В., Захарова Е. Я., Титова В. А. Информационные системы : теоретические предпосылки к построению. 2-е изд. Новосибирск: НГТУ, 2014. 283 с.

19. Отчет компании «We are social» о использовании сети интернет населением мира // ООО «Флекс Медиа»: [сайт]. [2016]. URL: http://flxmd.by/blog/otchet-we-are-social-42-naseleniya-mira-ispolzuyut/ (дата обращения: 03.07.2016).

20. Отчет Минтруда об увеличении коэффициента рождаемости в России // «Russia today»: [сайт]. [2016]. URL: https://russian.rt.com/article/323860-mintrud-v-rossii-uvelichilsya-koefficent-rozhdaemosti- (дата обращения: 20.07.2016).

21. Официальная демографическая статистика // «Росстат»: [сайт]. [20016]. URL: http://www.gks.ru/wps/wcm/connect/rosstat_main/rosstat/ru/statisti

cs/population/demography/ (дата обращения: 21.07.2016).

22. Официальная документация по системе MODX // «MODX Docs»: [сайт]. [2016]. URL: https://docs.modx.com/revolution/2.x/getting-started/server-requirements (дата обращения: 14.08.2016).

23. Персианов В.В., Логвинова Е.И. Информационные системы: учебно-методическое пособие. М.: Директ-Медиа, 2016. 191 с.

24. Рейтинг CMS по версии iTrack // Компания «iTrack»: [сайт]. [2016]. URL: http://www.itrack.ru/research/cmsrate/ (дата обращения: 08.08.2016).

25. Савельев А.О., Алексеев А.А. HTML5. Основы клиентской разработки. М.: Национальный Открытый Университет «ИНТУИТ», 2016. 272 с.

26. Савельева Н.В. Язык программирования PHP. М.: Национальный Открытый Университет «ИНТУИТ», 2016. 330 с.

27. Социологическое исследование российских врачей, акушерок и рожениц // «Акушерство сегодня»: [сайт]. [2016]. URL: http://www.midwifery.ru/today/opros_woman.htm (дата обращения: 04.08.2016).

28. Сычев А.В. Перспективные технологии и языки веб-разработки. М.: Национальный Открытый Университет «ИНТУИТ», 2016. 494 с.

29. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. СПб.: Питер, 2012. 960 с.

30. Уткин В.Б., Балдин К.В. Информационные системы в экономике. М.: Издательский центр «Академия», 2012. 288 с.

31. Федеральный закон "Об информации, информатизации и защите информации" от 20.02.1995 N 24-ФЗ // «КонсультантПлюс»: [сайт]. [2016]. URL.: http://www.consultant.ru/document/cons_doc_LAW_5887/ (дата обращения: 07.07.2016).

32. Федеральный закон "Об участии в международном информационном обмене" от 04.07.1996 N 85-ФЗ // «КонсультантПлюс»: [сайт]. [2016]. URL.: http://www.consultant.ru/document/cons_doc_LAW_10929/ (дата обра-

щения: 10.07.2016).

33. Флэнаган Д. JavaScript. Подробное руководство. 6-е изд. СПб.: Символ-Плюс, 2012. 1080 с.

34. Царев Р.Ю., Пупков А.Н., Самарин В.В., Мыльникова Е.В. Информатика и программирование: учебное пособие. Красноярск: Сиб. федер. ун-т, 2014. 132 с.

35. Царев Р.Ю., Пупков А.Н., Самарин Е.В. Теоретические основы информатики. Красноярск: Сиб. федер. ун-т, 2015. 176 с.